Computer Organization and Design

Chapter 2

The Language of the Computer

Sections 2.1 – 2.3, 2.5 – 2.8, 2.10 – 2.12

Computer Languages

- HLL (High Level Languages)
  - Ex. Java, C, C++, FORTRAN, COBOL, BASIC, Python
  - Designed to allow programmers to relate problem solving to human ways of thinking
  - Statements are constructed that define one or more operations
  - Words and symbols are used to define logic or mathematical processes
  - Most HLLs require a specific syntax in order for the compiler/interpreter to create the executable form of the program

Computer Languages

- Approach to HLL programming is based on a paradigm:
  - Structured (procedural) programming
    - aimed at improving the clarity, quality, and development time of a computer program by making extensive use of procedures, functions, block structures, for and while loops
  - Object-oriented programming
    - based on the concept of "objects", which may contain data, in the form of fields, often known as attributes; and code, in the form of procedures, often known as methods
  - Functional programming
    - treats computation as the evaluation of mathematical functions and avoids changing-state and mutable data
    - programming is done with expressions or declarations instead of statements (sometimes highly symbolic)

Hardware Language

- Computer hardware can only respond to native binary machine language
- Machine language is specific to the architecture of the system
  - Direct correlation between a machine instruction and the underlying hardware that implements it
- All HLL code must be translated to machine language for the hardware to run it

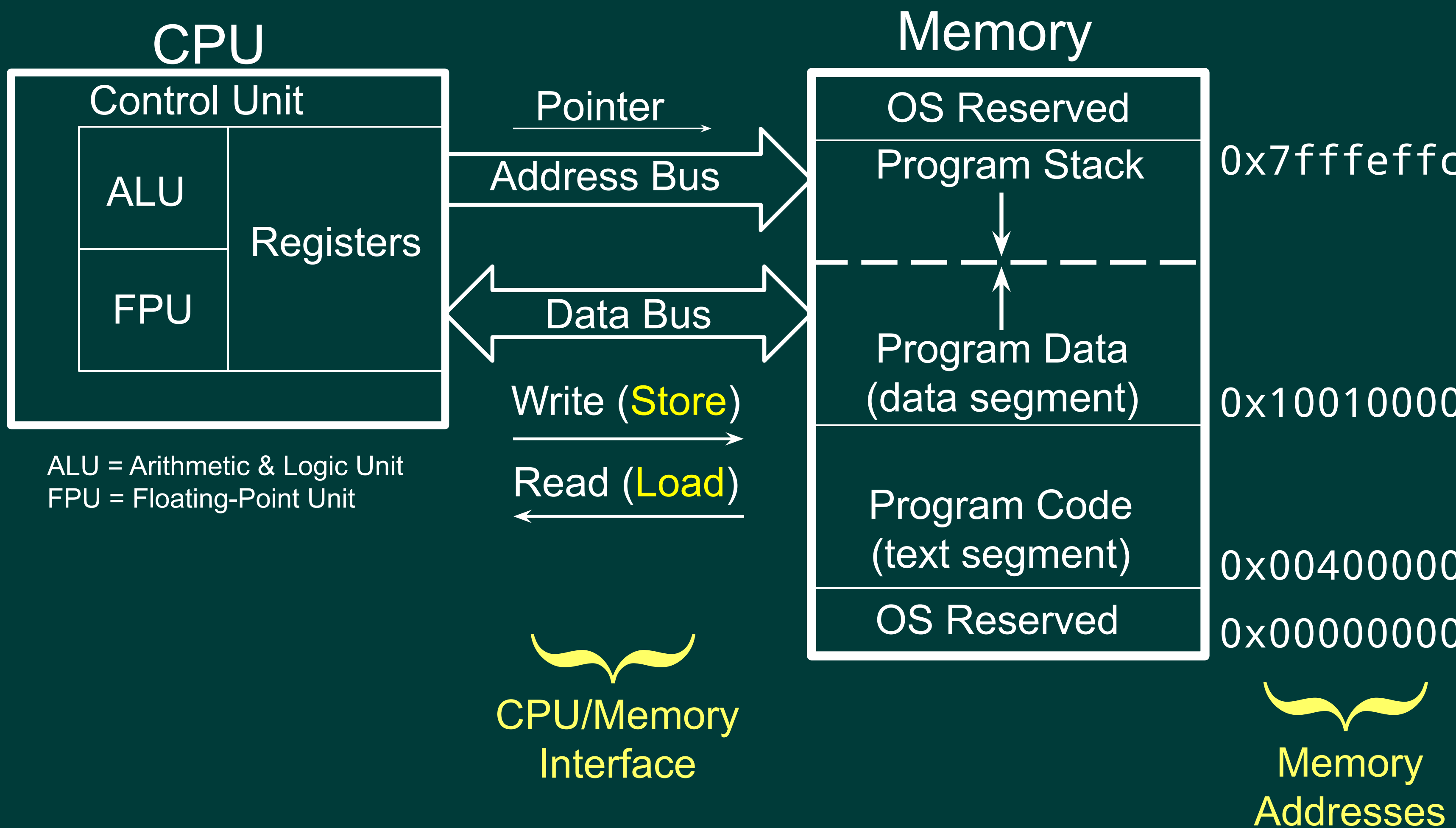
Instruction Set

- Instruction: specific operation to be performed in the hardware
- Instruction set: set of all instructions (all possible operations that can be performed)
- Instruction sets of different computers are
  - Similar because
    - All computers are built using hardware technologies based on similar underlying principles
    - There are few basic operations that all machines must provide
  - Different because
    - Internal structure of different computers can vary
      - Microarchitecture is different
      - Microarchitecture defines the implementation of the ISA
    - Level of functions provided also vary
- Instruction set is inherent to hardware design

RISC-V

- RISC-V is an open standard instruction set architecture (ISA) based on established reduced instruction set computer (RISC) principles
- Development began at UC Berkeley in 2010
- RISC-V is a newer architecture developed to solve many issues of prior architectures
  - open-source: managed by non-profit RISC-V Foundation
  - no royalties for use
  - flexible: suitable for embedded systems through high-performance systems
  - accommodate all implementation technologies (FPGA, ASIC)
  - supports specialization for customized applications
  - designed for non-obsolescence
- RISC-V Foundation members include Google, NVIDIA, Samsung, Qualcomm, IBM and many others

RISC-V Programming Model
(Load/Store Architecture)

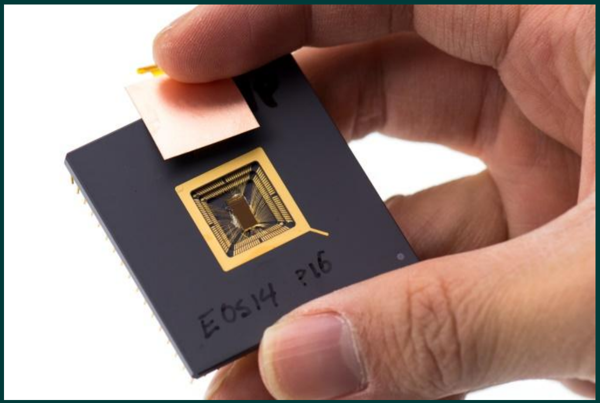


RSIC-V Architecture

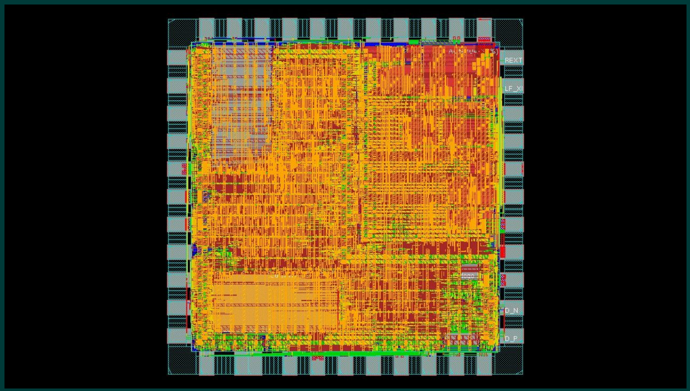
- RISC-V defines 32- and 64-bit base architectures with extensions
  - RV32I (32-bit integer base)
  - RV64I (64-bit integer base)
- Standard extensions
  - M - Integer Multiplication and Division
  - A - Atomic Instructions
  - F - Single-Precision Floating-Point
  - D - Double-Precision Floating-Point
  - Q - Quad-Precision Floating-Point
- This course will use RV64I with M, F and D extensions

RV64I

- Memory address values are 64 bits
  - RARS program only supports 32 bit addresses
  - Data memory limited to 127 KiB
- Largest integer size is 64 bits
- Instruction size is 32 bits
- Internal CPU storage registers are 64 bits



RISC-V prototype processor (2013)



RISC-V processor (SiFive single core)