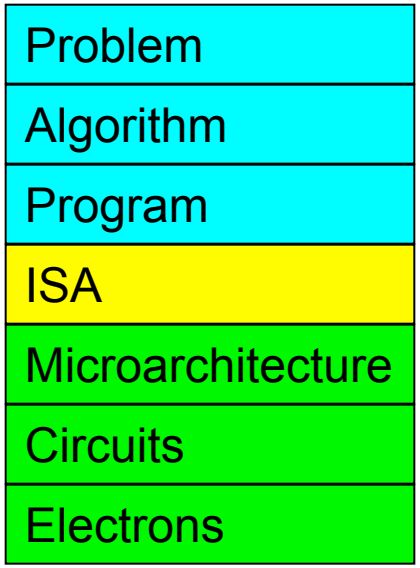
CS161 – Design and Architecture of Computer Systems

ISA Overview

UNIVERSITY OF CALIFORNIA, RIVERSIDE

ISA vs. Microarchitecture

- ISA
  - Agreed upon interface between software and hardware
    - SW/compiler assumes, HW promises
  - What the software writer needs to know to write and debug system/user programs
- Microarchitecture
  - Specific implementation of an ISA
  - Not visible to the software
- Microprocessor
  - **ISA, uarch**, circuits
  - "Architecture" = ISA + microarchitecture



ISA

- Instructions
  - Opcodes, Addressing Modes, Data Types
  - Instruction Types and Formats
  - Registers, Condition Codes
- Memory
  - Address space, Addressability, Alignment
  - Virtual memory management
- I/O: memory-mapped vs. instr.

Microarchitecture

- Implementation of the ISA under specific design constraints and goals
- Anything done in hardware without exposure to software
  - Pipelining
  - In-order versus out-of-order instruction execution
  - Memory access scheduling policy
  - Superscalar processing (multiple instruction issue?)
  - Caching? Levels, size, associativity, replacement policy
  - Prefetching?

Property of ISA vs. Uarch?

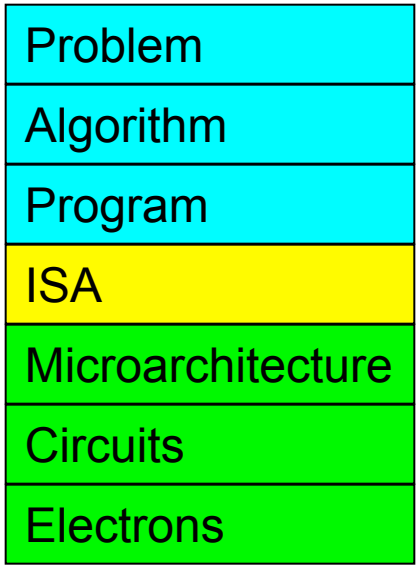
- ADD instruction's opcode
- Number of general purpose registers
- Number of ports to the register file
- Number of cycles to execute the MUL instruction

- Remember
  - Microarchitecture: Implementation of the ISA under specific design constraints and goals

Design Point

- A set of design considerations and their importance
  - leads to tradeoffs in both ISA and uarch

- Considerations
  - Cost
  - Performance
  - Maximum power consumption
  - Energy consumption (battery life)
  - Reliability and Correctness



- Design point determined by the "Problem" space (application space), the intended users/*market*

MIPS ISA

MIPS arithmetic

- All instructions have 3 operands
- Operand order is fixed (destination first)

Example:

C code: `a = b + c`

MIPS 'code': `add a, b, c`

"The natural number of operands for an operation like addition is three…requiring every instruction to have exactly three operands, no more and no less, conforms to the philosophy of keeping the hardware simple"

MIPS arithmetic

- Design Principle: simplicity favors regularity.
- Of course this complicates some things...

C code: `a = b + c + d;`

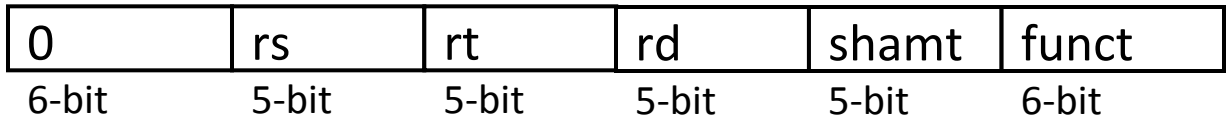
MIPS code:

```
add a, b, c
add a, a, d
```

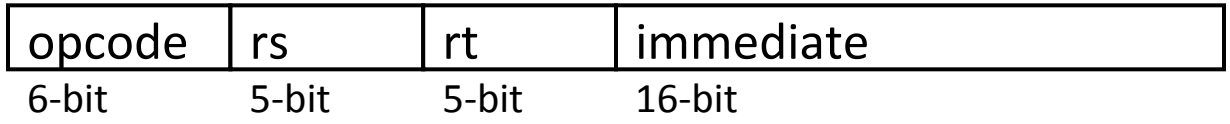
- Operands must be registers, only 32 registers provided
- Each register contains 32 bits

- Design Principle: smaller is faster. Why?

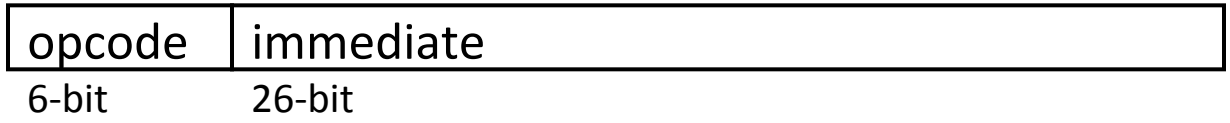
MIPS Instructions Format



R-type



I-type



J-type