Lecture 24
July 7

ISA's

- Operand addressing
  - How many operands?
- Address architecture
  - What type of operands are allowed?

Addressing Modes

- How are operands interpreted?
- Implied mode
  - Node address field given
  - Operand is specified as port of the definition of the opcode
  - E.g. Stack0based operations
- Immediate mode
  - The operand itself is specified as part of the instruction
  - E.g. ADD#7 ACC←ACC+7
- Register mode
  - The address specifies a register
    - Use value from that register
  - E.g. ADD R7 ACC←ACC+R7
- Register-indirect mode
  - The register specifies a memory address.
  - E.g. ADD(R7) ACC←ACC+M[R7]
  - LD R0, (R7) R0 M0[R7]
- Direct addressing mode
  - The operand specifies a memory address
  - E.g. ADD 7 ACC ←ACC+M[7]
- Indirect addressing mode
  - The address is a memory address
    - Use the value there
    - As a memory address for the operand
  - E.g. ADD[7] ACC←ACC+M[M[7]]
- Relative addressing
  - Gives a memory address relative to the PC
  - E.g. ADD#7 ACC ←M[PC+7]
- Indexed addressing
  - Gives a memory address, relative to an index register
  - e.g. ADD7(X) ACC←ACC+M[X+7]
  - the (X) could be a special-purpose indexing-register or a general purpose register
- many other addressing mode are possible

RISC and CSIS

- When an instruction set arch. Is defined, we have to chose the operand accessibility, modes, reg. file size… etc.
- CISC and RISC represent two types of ISA
- CSIS:
    - "complex ISA" (not pipelined)
    - instructions can take more than one cycle
    - lots of addressing modes
    - most instructions can do memory access (memory-memory or memory-register)
    - instructions can have different lengths
        - 1 word, 2 words, etc.
        - usually depends on addressing mode
- RISC
    - "reduced ISA"
    - instructions all take a single cycle to complete
    - usually pipelined
    - few addressing modes
    - few instructions can do memory access (load-store)
    - all the instructions are the same size
- Real processors range between pure RISC and pure CSIC?
    - Lots of choice for assembly programmers
        - Many addressing modes
        - Lots of instructions that do a lot

Why RISC?

- Easier fro compliers to work with
    - Smaller instructions are easier to recognize
- Simpler circuitry to complement
    - Cheaper to produce
- Can have a faster clock
    - Simpler circuit→shorter propagation delay
- Pipeline
    - Faster
- Assembly programs in RISC tend to be longer
- RISC architectures usually execute code for a given task faster