Lecture 5, Feb 6, 2024

Branching

- A branch instruction is an instruction that modifies the program counter, changing the flow of code execution
- We will make a distinction between *branch* and *jump*:
  - *Branch* has a limited range of movement, usually relative to the current program counter
  - *Jump* can access the full range of memory and set the program counter anywhere
- A *direct branch* is the simplest variant that directly loads a value into the PC
  - e.g. `BRA 0xF5`
  - Typically a simplified or a low-cost version of a full jump since it addresses a smaller range of memory
  - Some platforms use it to jump to interrupt code
  - Essentially a `goto`
- A *relative branch* increments or decrements the PC by some value, relative to the current PC
  - e.g. `BRR 0x05`
  - The offset can be negative to jump back to earlier code
  - Note the offset usually has less bit width than the address, so we cannot access addresses that are too far away
  - We also need to perform sign extension when adding this offset to a PC with larger bit width
- Relative branching allows relocatable code, i.e. we can move a chunk of code anywhere and it will execute the same since it does not use any hard-coded addresses within itself
  - This allows the code to be compiled once and dynamically linked
  - Also allows for self-modifying code, which allows the existence of bootloaders, OSes, etc
- To calculate the offset for relative branching, subtract the current PC (after reading the branch instruction) from the destination PC
  - The offset is the address of the destination instruction, minus the address of the branch, minus the size of the branch instruction
- A *jump* simply loads the PC with the direct value in the instruction
  - This allows accessing the entire memory range of the processor
  - Typically we don't do math with jump instructions
- *Conditional* branches and jumps only occur if a condition is true
  - The test usually involves a subtraction and testing if the result is positive/negative/zero
  - The conditional branch usually relies on values in the CCR (status register)
    - This means we can branch off not just subtractions but anything that sets the CCR
  - Depending in ISA we may have branching instructions for result equal to zero, positive/negative, overflow, etc
  - Some ISAs offer a compare instruction, which is a specialized subtraction that may have benefits such as not modifying the operands, restoring the original CCR bits after branching, etc