CS 271 Lecture #2

How computer hardware works

Preliminaries

- Inside a computer, information is represented electrically
  - Smallest unit of information is a switch
    - May be on or off
- We often represent “off” as 0 and “on” as 1, so a single switch represents a binary digit, and is called a “**bit**”.
- Different combinations of switches represent different information.
  - A group of 8 bits is called a **byte**

Main parts of the CISC diagram (Complex Instruction Set Computer)

- **Peripheral Devices**: External devices:
  - Store/retrieve data (non-volatile storage)
  - Convert data between human-readable and machine readable forms.
- **I/O Unit**: Hardware/software functions:
  - Communicate between CPU/Memory and peripheral devices
- **Main Memory Unit**: Cells with addresses:
  - Store programs and data currently being used by the CPU (volatile storage)
- **CPU**: Central Processing Unit:
  - Execute machine instructions

Components / Terms

- **Bus**: parallel "wires" for transferring a set of electrical signals simultaneously
  - Internal: Transfers signals among CPU components
  - Control: Carries signals for memory and I/O operations
  - Address: Links to specific memory locations
  - Data: Carries data CPU ⇔ memory
- **Register**: fast local memory inside the CPU
- **ALU**: Arithmetic/Logic Unit
- **Microprogram**: sequence of micro-instructions (implemented in hardware) required to execute a machine instruction
- **Micromemory**: the actual hardware circuits that implement the machine instructions as microprograms

Registers

- **Control**: dictates current state of the machine
- **Status**: indicates status of operation (error, overflow, etc.)
- **MAR**: Memory Address Register (holds address of memory location currently referenced)
- **MDR**: Memory Data Register: holds data being sent to or retrieved from the memory address in the MAR
- **IP**: Instruction Pointer (holds memory address of next instruction)
- **IR**: Instruction Register (holds current machine instruction)
- **Operand_1**, **Operand_2**, **Result**: ALU registers (for calculations and comparisons)
- **General**: fast temporary storage

Cache

- Cache: an area of comparatively fast temporary storage for information copied from slower storage.
  - Examples:
    - Program instructions are moved from secondary storage to main memory, so they can be accessed more quickly
    - Data is moved from main memory to a CPU register, so it can be accessed instantaneously.
- Caching takes places at several levels in a computer system.
  - More later on caching

Machine instructions

- Each computer architecture provides a set of machine-level instructions
  - Instruction Set Architecture (**ISA**)
  - Specific to one particular architecture
- Like everything inside a computer, machine instructions are implemented electrically
  - Micro-instructions set the switches in the control register

Real computers …

- … use the “stored program” concept
  - VonNeumann architecture
    - Program is stored in memory, and is executed under the control of the operating system
- … operate using an Instruction Execution Cycle

Instruction Execution Cycle

1. Fetch next instruction (at address in IP) into IR.
2. Increment IP to point to next instruction.
3. Decode instruction in IR
4. If instruction requires memory access,
   A. Determine memory address.
   B. Fetch operand from memory into a CPU register, or send operand from a CPU register to memory.
5. Execute micro-program for instruction
6. Go to step 1 (unless the “halt” instruction has been executed)

Note: default execution is sequential

Discussion question #4:

- In the Instruction Execution Cycle, why is it important to change the instruction pointer in step 2 ? Wouldn't it work just as well to change it after step 5 ?

Example CISC Instruction

```
ADD     R1, mem1        ;Example assembly language instruction
```

Meaning: Add value in memory location mem1 to value in register R1)

Example ADD Microprogram

(each micro-instruction executes in one clock cycle)

1. Copy contents of R1 to ALU Operand_1
2. Move address mem1 to MAR
3. Signal memory fetch (gets contents of memory address currently in MAR into MDR)
4. Copy contents of MDR into ALU Operand_2
5. Signal ALU addition
6. Set Status Register and Copy contents of ALU Result to register R1

Things get complicated …

- Even in the simplest architectures
  - Bus Arbitration required
  - CPU scheduling required
- As architectures become more complex
  - multi-processor coordination required
  - cache management required
- Etc. …