Programming From the Ground Up - Exercise Solutions

Michael Crawshaw

December 15, 2020

Chapter 2 - Computer Architecture

Know the Concepts

1. Describe the fetch execute cycle.
   Answer: The fetch execute cycle is the process of operations performed by the hardware to carry out an instruction. The CPU fetches the value pointed to by the program counter, then the instruction decoder coverts this value into both an instruction and the memory locations involved in the instruction. The relevant data is fetched from the corresponding memory locations through the data bus and placed into register(s), then the operation is executed by the arithmetic and logic unit and the result is placed in a register or the appropriate memory location.

2. What is a register? How would computation be more difficult without registers?
   Answer: Registers are memory locations within the CPU distinguished from main memory and used for computation. Without registers, computation would have to be performed directly on data in main memory, which would be both cumbersome and slow.

3. How do you represent numbers larger than 255?
   Answer: Numbers larger than 255 are represented by concatenating the values stored in multiple bytes.

4. How big are the registers on the machines we will be using?
   Answer: On the machines we will be using, the registers will contain 4 bytes.

5. How does a computer know how to interpret a given byte or set of bytes in memory.
   Answer: How a piece of memory should be interpreted is specified to the computer by individual instructions. An instruction may specify for the value at a given address to be added to another value (interpreted as data) or the program counter may point to a given address (interpreted as executable).

6. What are the addressing modes and what are they used for?

Answer: The various addressing modes are immediate mode, register addressing mode, direct addressing mode, indexed addressing mode, indirect addressing mode, and base pointer addressing mode. Each mode of is a different way of specifying the data to be used in an instruction, either by providing that data directly, providing the address to that data, etc.

7. What does the instruction pointer do?
   Answer: The instruction pointer points to the location in memory that holds the next instruction for the processor to execute.

Use the Concepts

1. What data would you use in an employee record? How would you lay it out in memory?
   Answer: For example, our employee record could contain a name (50 bytes), an employee ID (one byte), and a yearly salary (four bytes). If the record starts at memory location `start`, then the name is stored at location `start`, the ID is stored at location `start + 50`, and the salary is stored at location `start + 51`.
2. If I had a pointer to the beginning of the employee record above, and wanted to access a particular piece of data inside of it, what addressing mode would I use?
   Answer: Since we have a pointer to the memory location we want to read from and an offset, we would use base pointer addressing mode.
3. In base pointer addressing mode, if you have a register holding the value 3122, and an offset of 20, what address would you be trying to access?
   Answer: 3142.
4. In indexed addressing mode, if the base address is 6512, the index register has a 5, and the multiplier is 4, what address would you be trying to access?
   Answer: 6532.
5. In indexed addressing mode, if the base address is 123472, the index register has a 0, and the multiplier is 4, what address would you be trying to access?
   Answer: 123472.
6. In indexed addressing mode, if the base address is 9123478, the index register has a 20, and the multiplier is 1, what address would you be trying to access?
   Answer: 9123498.

Going Further

1. What is the minimum number of addressing modes needed for computation?
   Answer: Technically, we could use only a single addressing mode (indirect addressing mode) without losing the ability to perform any computations that we can with all addressing modes. This is because any other addressing mode can be simulated by

indirect addressing mode; we just need to store the desired memory location in a register.

2. Why include addressing modes that aren't strictly needed?
   Answer: The addressing modes are all included for convenience. Sometimes we can save register space (when using immediate addressing), other times it is simply easier to have options besides only using pointers in registers.

3. Research and then describe how pipelining (or one of the other complicating factors) affects the fetch-execute cycle.
   Answer: Pipelining is a method for parallelizing the individual operations in the fetch-execute cycle to maximize utilization of CPU resources. Since the execution of a single instruction requires several steps (fetch instruction, decode instruction, execute instruction, etc), the first step of one instruction can be executed at the same time as the second step of another instruction. For example, suppose we have two instructions to execute with pipelining. On the first timestep, we would fetch the first instruction. On the second timestep, we would both decode the first instruction and fetch the second instruction. On the third step, we would execute the first instruction and decode the second instruction, and so on. This parallelization allows for better resource utilization and faster execution times.

4. Research and then describe the tradeoffs between fixed-length instructions and variable-length instructions.
   Answer: Fixed-length instructions (RISC for Reduced Instruction Set Computer) create consistent instruction lengths, which are easier to handle and execute than variable-length instructions. However, the complexity of a single instruction is inherently limited by its size, so a fixed-length instruction puts a cap on the complexity of each instruction. Some examples of RISC systems are ARM and SPARC. On the other hand, variable-length instructions (CISC for Complex Instuction Set Computer) require more work to handle and execute, but they allow for more flexibility with longer instructions as well as space-saving for short instructions. Some examples of CISC systems are Intel and M68000.