Chapter 2

Significance

Computer organization is essentially a study of the processor, memory hierarchy, I/O system and bus and how these components work together to execute program instructions. It is perhaps not necessarily for the IT student to understand the details here. However, it is essential that the IT student be able to deal with the consequences of computer organization: being able to evaluate processor performance, understanding the memory hierarchy, knowing about the interrupt system, understanding concepts like computer's word size, knowing what a pipeline is, etc. This will allow the IT person to not only make informed decisions when buying or upgrading systems, but many of these concepts will help the IT person understand the performance of the computer system. This chapter covers several organization topics but it also introduces related topics. After the lengthy discussion of the structure of the computer and the fetch-execute cycle, we visit the CPU in some detail, followed by the memory hierarchy and then the I/O system. The section on the I/O system discusses the peripheral devices and such concepts as human-computer interaction (HCI) and plug-and-play. The last portion of this chapter more concretely illustrates the components of the computer and how you can assemble them. The first lab in the lab manual covers PC assembly.

Chapter outline

I. Introduction – the IPOS cycle
II. Structure of the computer
    a. Digital devices
    b. The bus components
III. Sample program
    a. C version
    b. Assembly version
    c. Brief glimpse of machine language version
IV. The fetch-execute process
    a. CPU components: ALU, control unit, registers of note
    b. 5-part fetch-execute cycle
    c. Executing the program
    d. Memory reads and memory writes
V. The CPU in detail
    a. Microcode
    b. System clock
    c. Evaluating a processor's performance
    d. Benchmark programs
VI. Memory hierarchy
    a. RAM, DRAM and ROM
    b. SRAM and on-chip and off-chip caches
    c. Virtual memory
VII. Input and output

a. Interactivity
b. HCI
    i. Ergonomics and repetitive stress injuries
    ii. Accessibility
    iii. Wearables
    iv. Virtual reality
c. Plug-and-play

VIII. Computer hardware

IX. Computer assembly

Chapter in more detail

This chapter is divided into four sets of material. First up is the fetch-execute cycle. This is described in the introduction and then the first three titled sections. The second of these titled sections includes a short sample program that is then used in the third section for reference. It is important that you step through the fetch-execute cycle to help the student truly understand what is going on, but do not be surprised if your students are confused by much of the detail. The program supplied in the text is a very simple one. Yet it may not illustrate all of the concepts clearly, so another example is provided below with the in-class activities.

Section 4 continues the coverage of computer organization by focusing on the CPU. We reinforce concepts covered in the previous sections and also discuss microcode, clock cycle speed, and attributes of a computer that impact processor performance. It is a great misnomer that clock cycle speed dictates the speed of the computer. It is important to emphasize this to the students. As your students may be involved in recommending computer purchases for their organization(s), they should understand the significance of this misnomer and how better to gage the computer's performance by looking at many characters (see page 33).

Section 5 covers the memory hierarchy which continues the coverage of computer organization. Here, we see the primary forms of memory and the impact that they have on each other, for instance, the significance of having on-chip and off-chip cache. We also introduce virtual memory which is a topic that will arise again in chapter 4.

In section 6, we explore the various forms of input and output. This moves us away from the lower level computer organization concepts into material that is probably easier for the student to understand. The discussion of wearables and virtual reality, which wraps up coverage of HCI, might go well with material covered later in the textbook such as the end of chapter 12 (which deals with the Internet) or chapter 16.

The final section of this chapter steps the reader through the various components of a computer and discusses how to assemble them. The lab maual's first lab involves PC construction. You may or may not have the hardware available to support this lab. If not, you should obtain the various pieces of equipment so that you can at least pass them out to the class for them to examine. Specifically, they should see a motherboard, CPU, banks of memory chips, and at least one type of disk drive. If you can also show them an open system unit, that would be beneficial.

In-class activities

- Demonstrate the various components of the computer system by showing the students actual components and passing them out to the class (if you have them available)
- Do a different fetch-execute example so that the students have two versions, the one in the text and the one from class. Here is an example:
    - C code:

```
x=n;
factorial=1;
while(x > 0) {
        factorial=factorial*x;
        x=x-1;
}
printf("%d\n", factorial);
```

    - Assembly code:

```
        Load n
        Store x             // x = n
        Load #1
        Store factorial     // factorial = 1
Top:    Load x              // top of the while loop, move x to AC
        Jle Done            // if x <= 0 jump to done
        Multiply factorial  // AC = AC * factorial
        Store factorial     // factorial gets new value
        Load x
        Subt #1
        Store x             // x = x - 1
        Jump Top
Done:   Load factorial
        Output 2049         // printf statement
```

    - Assume the program is loaded into location 1,000,000, n, x and factorial are stored in locations 1,000,014, 1,000,015 and 1,000,016 respectively. Top is at location 1,000,004 and Done is at location 1,000,012. The PC will start out as 1,000,000.
    - We start with PC = 1,000,000 and fetch Load n, incrementing the PC to 1,000,001. We decode this instruction and execute it, causing the datum n (at memory location 1,000,014) to be loaded into the accumulator.
    - At 1,000,001, we fetch Store x and increment the PC. We decode this instruction and execute it by sending the value in the AC, a write signal, and the address of x (1,000,015) to memory. This stores the value from n into x.
    - …
    - Skipping ahead two instructions, we fetch Load x and execute it placing the current value of x in the accumulator.
    - Next, we fetch Jle Done and increment the PC to 1,000,006. Here, we test the accumulator. If the value is <= 0, we change the PC to the location Done (which

is memory location 1,000,012. So, when this instruction is done, the PC is either 1,000,006 or 1,000,012.

- Assuming it is 1,000,006, we fetch Multiply factorial. This requires that we fetch factorial from memory (location 1,000,016). Executing this instruction causes the multiplier to multiply together the value in the AC and the value fetched from memory (factorial), placing AC * factorial in the AC. The next instruction, when executed, causes us to move this product back into the variable factorial in memory (1,000,016).