MODULE 14 - BASIC COMPUTER ARCHITECTURE

OVERVIEW:

The modern digital computer has existed for less than 50 years. During this time, much fine tuning of the design has occurred to speed up computers and to make them more efficient. Even so, the Basic Architecture has not changed significantly from the first computers that filled entire building to the small notebook computers we use today. The main changes have been in how fast they operate, how much memory they have, the resolution of the display, and the level of sophistication of the software. This module will look at the microcomputer chip architecture, and the computer bus architecture, and look at the various peripherals that are used allow the computer to interact with the world. We will examine how programs are written and how they make a computer do something useful.

CONCEPT 14.1: BASIC COMPUTER BUS ARCHITECTURE

The basic design used in the first microprocessors is still used with some additions to make it faster and to give it the ability to handle more data. To understand computer architecture, we need to look at two structures, the external or bus architecture, and the internal or chip architecture. Figure 14.1 shows the most simple bus architecture reduced down to a Central Process Unit (CPU), Its there main busses (DATA, ADDRESS, and CONTROL), its memory (RAM & ROM), and its interface with the outside world (The Input/Output PORTS).

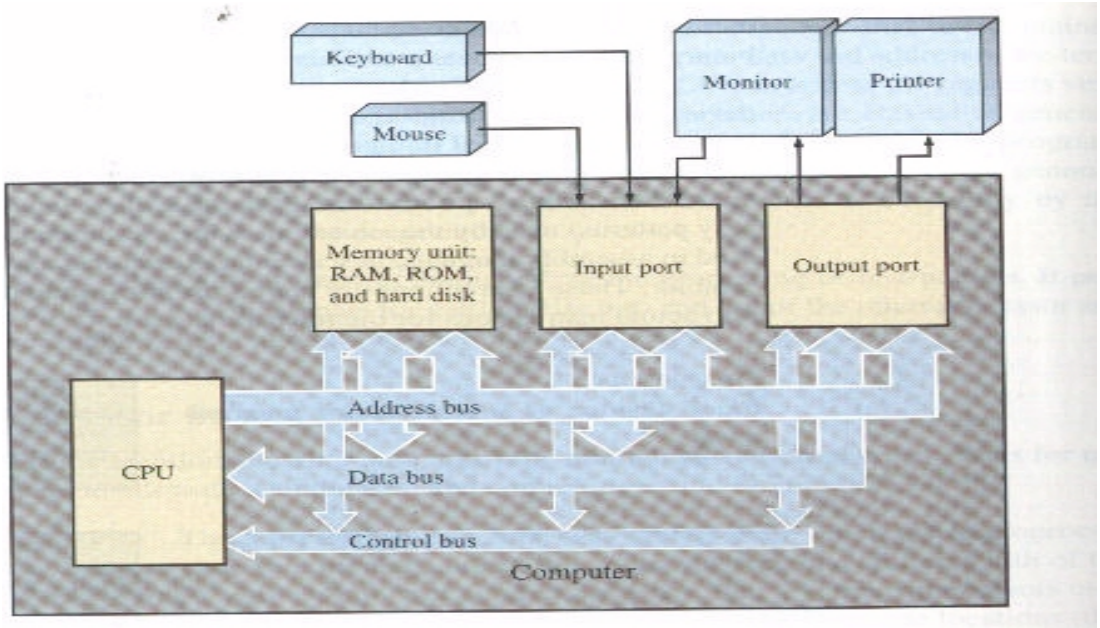

FIGURE 14.1: COMPUTER BUS ARCHITECTURE

CONCEPT 14.2: BASIC COMPUTER CHIP ARCHITECTURE

Figure 2 shows the basic CPU chip architecture. It has Buffers that interface it to the three outside busses (ADDRESS BUS BUFFER, DATA BUS BUFFER, and CONTROL BUS BUFFER), and internal bus that carries internal data (MICRO BUS), an internal control ROM (MICRO CONTROLLER), an Arithmetic - Logic Unit (ALU), and internal data storage registers (ACCUMULATORS, COUNTERS, POINTERS, FLAGS).

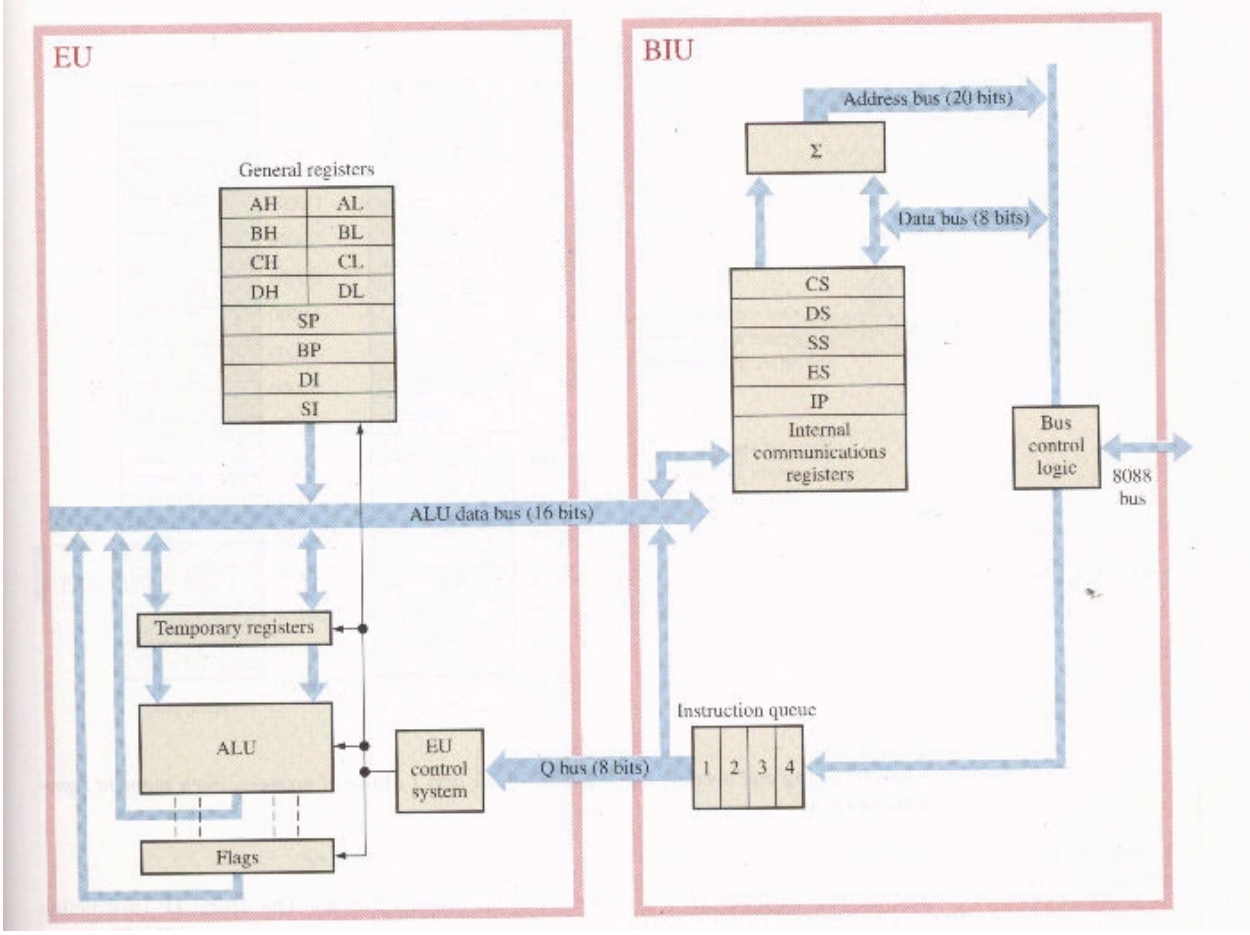

FIGURE 14.2: BASIC COMPUTER CHIP OR CENTRAL PROCESS UNIT ARCHITECTURE

CONCEPT 14.3: THE "FETCH – EXECUTE" CYCLE

Since the internal Micro Controller is a ROM, each instruction that comes into the CPU acts like an address to direct the ROM Controller to execute a specific instruction.

It does it by turning on and off different registers to pass data around inside using the Micro Bus, and then it controls what is done to the data by intern manipulating the ALU. When it finishes with an instruction, it increments its address counter and points to the next instruction. It fetches the next instruction and then proceeds to execute it in the same manner. A program is just a sequence of instructions that logically progress through the steps a person might take to add up a string of numbers or to print a document.

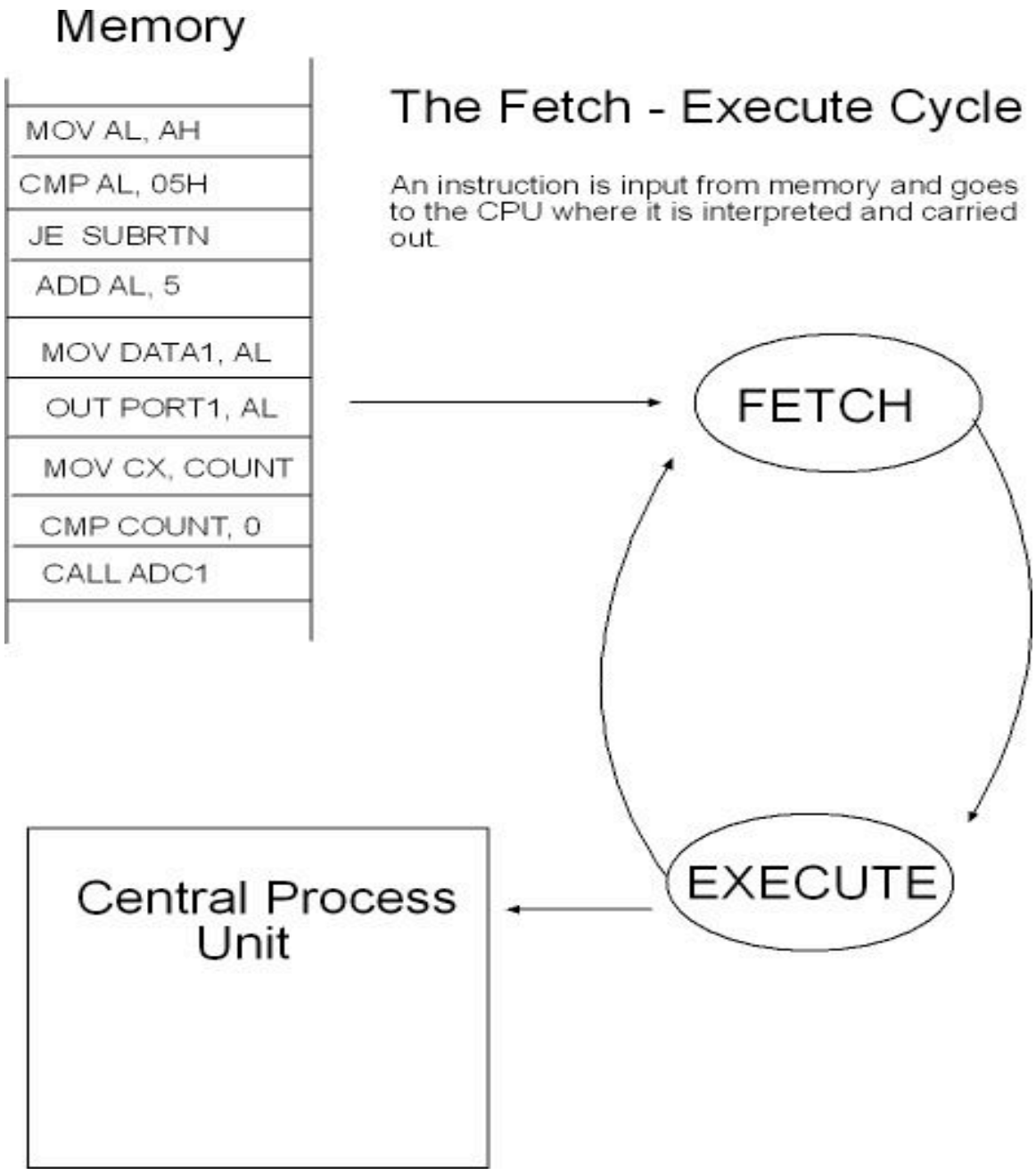

FIGURE 14.3: THE FETCH – EXECUTE CYCLE

Because the computer is so fast, it can repeat boring, tedious operations millions of times for us and never get tired. It can type and store strings of data, step by step process numbers, send our work out to a printer on letter at a time, or even draw a beautiful picture on a CRT Display one little color dot or pixel at a time. A programmer took the time and effort to teach it to do it once and after that it stored the process and repeats it each time we ask it to perform.

If electrical noise causes the data to change even by one bit, the result can be a computer crash. The computer can lose count of where it is executing code and literally jump to a different location in memory. When this happens the computer generally ceases to communicate with our display, mouse or keyboard and either locks up or prints something strange on the display screen. The computer must be reset. Resetting the computer starts the program over. It "boots up", or relearns the basics of communicating with the keyboard, displaying data on the screen and accessing its memory devices. The code necessary for the basic functions we just described is called the Basic Input Output System or BIOS for short. It resides in ROM memory because ROM memory is non-volatile. These vital programs are called "Firmware" programs. That means it will not be effected by turning the computer on or off, and will not be altered by electrical noise. This is very important. It is like the basic life functions we have to keep our heart going or make us breath while we sleep.

Programs like Word Processors, Video Games, and the like are loaded into RAM or soft memory. If the power glitches, these programs crash. The worst scenario is that you might lose a letter you are typing. This is why such programs are called "Software". Other than their volatility, Firmware and Software are exactly alike in the manner they are executed by the computer. When you load RAM with a Software program, you begin executing that program by changing the address counter to the same address where you loaded the program. The program then begins by following its instructions step by step, instruction by instruction gradually doing what the program was designed to accomplish. In subsequent classes, we learn how to logically design such programs using a knowledge of how the computer stores and processes data. No magic takes place. It just looks that way. In reality, computers are boring in their simplicity and tedious in their one step at a time approach to doing things.