CDAC 3331: Introduction to Microprocessor Systems
Comments for the First Day of Class

Dr. Michael VanHilst

The syllabus is on my web site: http://www.cse.fau.edu/~mike. Your assignment for Thursday is to read chapter 1 of the book.

This course was designed by Bassem Alhalabi – he creates the lab assignments that you will do. (I will create your quizzes and exam.) Bassem is famous for the way he teaches this class. His section in Boca is always full. It is worth checking his web site periodically for useful information: http://www.cse/fau.edu/~bassem/courses.htm

I apologize for being away on the first day. I am at MIT in Boston. I will be back on Thursday. My office hours are before class on Tuesday and after class on Thursday. But I am usually available before and after every class. My email is mike@cse.fau.edu. Please note, I do not read my email on weekends.

Who am I? This is my third year of teaching at FAU. Before becoming a teacher I did software research at HP labs in Palo Alto, California. I was an undergrad at MIT in the 70s, worked in astronomy (at Harvard) as a programmer and hardware specialist in the 80s where I used microprocessors a lot, and did a stint in Paris for a year, where I programmed the chip you will be using. I went back to school when I was 37 and got my PhD at the University of Washington, in Seattle at the age of 44.

Who are you? Since I am not here to hear your stories, I will ask this on Thursday.

What does it take to succeed in this class? You are expected to learn how to program in a type of language called assembly, and to manipulate bits, binary, octal, and hex numbers, and numbers that give the locations where things can be found. Programming in assembly is not like programming in other languages. It's more like solving a puzzle. You will have to think like a machine. Many things are backwards from what you have learned in C or C++. The book is dense with very few examples. We will only use the first 133 pages or so. Some students have a hard time learning how to think in this strange way. In the beginning you will all find it difficult, but most of you will catch on, gradually, in pieces.

Your grade is based on what you have learned, not on effort. If you have trouble, talk to me! It only gets worse if you fall behind. I will fail you if you haven't succeeded. I also teach Senior Design. I'd rather fail you now, than fail you then – after you have spent all that money and time.

The five lab assignments in this class are programming assignments. You can do the assignments at home. The book comes with tools to do the work on a PC. But you must demonstrate them in the lab to get graded. You can also come in to the lab to get help.

The course also has five quizzes. The later quizzes will require you to program under pressure. Bassim posts practice quizzes to help you prepare. Most likely I will give a sixth quiz and then drop your lowest quiz grade. I may give additional assignments and practice quizzes, which I critique and give back, if I find that you need more feedback loops to get up to speed on the material.

What is a microprocessor?

A microprocessor is the logical part of a computer contained within a piece of plastic, with legs hanging out for electrical connection. A microprocessor is dumb. It can only do a few simple operations. But it can do them blazingly fast. (As an insult at MIT, we used to say "You think like a computer, only slower.")

The first computers were mechanical calculating machines with gears and levers. The first modern electronic devices that we now call computers used vacuum tubes (things that look like light bulbs and work like transistors). A computer with the processing power of your wrist watch filled most of a room. The first vacuum tube computer, the Collossus, was built by the British in 1943 to break German cypher codes. But its existence was a secret for a long time. The first American all electronic computer, the ENIAC, was built by the University of Pennsylvania, with 17000 vacuum tubes, and delivered in 1946. It filled a 30 by 50 foot room. Its inventors went on to build the first UNIVAC computer in 1951.

The first digital computers were used to replace roomfuls of women who compute tables of numbers for artillery trajectories (given wind and distance) and tables of actuarial statistics for insurance companies (on which they base their fees). My mother was one of those women computing insurance company tables after the war, and joined UNIVAC as an early programmer in the mid 50s. She stayed at Univac writing compilers until retiring in 1982.

Vacuum tube computers are called 1st generation (1945-1955). The transistor was invented in 1947. Transistor computers are called 2nd generation (1956-1963). Then the transistors were combined in logic chips like those you used in Logic Design class. Logic chip computers are called 3rd generation (1964-1971). When I started in the field, computers required many large boards filled with such chips. The chips often failed – my first computer job was to find the misbehaving chips. Microprocessor based computers are called 4th generation.

Intel first combined all the logic elements on a single chip in 1971. The Intel 4004 had 2300 transistors and a 4 bit bus (meaning it could only move numbers with values up to 15 on or off the chip in a single step). But it could perform 60000 of its simple instructions per second. The story goes that a Japanese company, named Busicom, asked Intel to provide it with chips for a calculator. Some Intel engineers realized that they could put all of the logic functionality on a single chip, with a more general set of programmable instructions. Busicom later sold the rights back to Intel.

A Timeline of Some Microprocessors
1971 Intel 4004, 4bit bus, 2300 transistors, 60kips – for Busicom
1972 Intel 8008, 8bit bus, 4300 transistors, 200kips
1974 Intel 8080, 8bit bus, 6000 transistors, 640kips (2Mhz clock)
1974 Motorola 6800, 8bit bus, 6800 transistors,
1978 Intel 8086, 16bit bus, 29000 transistors, 330kips (4.7Mhz clock)
1979 Motorola 68000, 16bit bus, 68000 transistors (hence the name!)
1997 Intel Pentium III, 32bit bus, 10000000 transistors, 500Mhz clock
2000 Intel Pentium 4, 32bit bus, 42000000 transistors, 1.5Ghz clock
2002 Intel Pentium 4, 32bit bus, 55000000 transistors, 3Ghz clock

A Timeline of Some Notable Computers that use Microprocessors
1981 IBM PC – used 8088, cheaper version of 8086 with 8bit bus.
1984 Apple Macintosh – used 68000
1996 Palm Pilot – used 68000 (DragonBall)

Why do we teach the 68000, instead of Intel or something more modern?
The 68000 has a very clean design and has a representative instruction set without too many peculiarities. This probably accounts for the long life of the original design. It is excellent for teaching the basics of assembly programming. By comparison Intel chips are harder to program because of their register and addressing peculiarities. Modern chips are more complicated to program because they employ many tricks to get more speed. You have to deal with instructions that are only partly executed, and different things happening at the same time, or even out of order.

Why should I learn about microprocessors when I can just use a PC?
PCs make up only a small fraction of the computers in the world. For example, a typical car has 32 computers. Your kitchen probably contains a handful of computers. Those computers aren't running Windows, and they have special needs you won't encounter in a PC. Even when you could use a PC for a product, a microprocessor will be way cheaper, smaller, and faster. Just look at the Sony PlayStation. The world of tomorrow will be filled with "computing appliances." (Ask me about doorknobs, and my friend and his camera if you haven't heard those stories.)

Why should I learn assembly?
You will understand a lot about computers when you are forced to think at their level. This understanding is invaluable for getting the most out of them. Also, to understand how computers interact with the outside world, you need to know what those pins are doing. In full disclosure, however, I will also tell you that developers shouldn't program in assembly that often. Modern C compilers can often beat the best hand coded assembly for efficiency, and the code is far more maintainable. (But the C code of a programmer who knows assembly will be better than that of one who does not.)

Note: the recorded lecture doesn't have the doorknob story or the MIT insult.(remind me)