General Overview of Class:

This class is an introduction to the low level operation of microprocessor-based computer systems. We will cover the operation of the Central Processor (CPU), Memory, and Input/Output (I/O) devices.

There will be four lectures per week and 2 lab session per week. Most of the labs are one session labs, however several are multi-session. (2 session and one 3 three session)

This class is primarily a programming class, however about a third is devoted to understanding the operation of the hardware and advanced architectural concepts.

The programming part of the class is primarily in assembly language using the Intel 8088 as the target processor, and MS-DOS as the target operating environment. Some programming will be done in C, and techniques for interfacing between C and assembly subroutines is covered. Many of the programming exercises in the labs are oriented around programming and controlling hardware peripheral devices.

The programming tools used will be Microsoft MASM as the assembler and Borland Turbo C as the C compiler. The Microsoft tools use PWB (Programmer’s Workbench) as the development environment (Editor and Make tools) with Codeview as the debugger. The Borland tools use the Turbo C IDE (Integrated Development Environment) as the Editor, Make tool and Debugger.

There will be a course web site on which class materials will be posted. These include: lecture notes, homework assignments, lab assignment materials, and various other handouts.

Prerequisites:

This class assumes prior experience with programming in the C language. It also assumes prior knowledge of basic logic design, both Combination and Sequential. Understanding of timing diagrams will be helpful.

Handouts:

- Course Syllabus and Lab Schedule
- Lab report standards
- Document on number systems and data formats
- Sample assembler listing will be posted on web site.

Evolution of Computers and the Development of the Microprocessor:

- Originally, computer was a job description. It meant someone whose job was to calculate things.
With the invention of programmable machines, the word changed to mean the machine that did the calculation.

Early computers required some rewiring to change operations. Eventually, the stored program concept was invented.

**Two principal architecture types**

Von Neuman architecture
- This has a single memory space for instructions and data

Harvard Architecture
- Separate memory space for instruction and data.

Mainframe computers

Mini Computers

Invention of Microprocessor -- Contains all of the functions of the CPU on a single integrated circuit.

<table>
<thead>
<tr>
<th>Model</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>8088</td>
<td>16 bits code and data, real mode operation</td>
</tr>
<tr>
<td>80286</td>
<td>16 bits code and data, real and protected mode operation</td>
</tr>
<tr>
<td>80386</td>
<td>32 bits code and data, real mode, virtual 86 mode and protected mode</td>
</tr>
<tr>
<td>80486</td>
<td>Integrates math co-processor functions with on same chip</td>
</tr>
</tbody>
</table>

Pentium
- Superscalar architecture

Parallel development of advanced CPU architectures resulting in Pentium, with integration of peripherals and memory onto chip, resulting in microcontrollers.

**System Block Diagram**

CPU – Central Processing Unit
- Controls operation of computer. Executes instructions. Generates timing to control operation of rest of system.
- Various kinds of Co-processors exist in some systems, which extend the instruction set of the main CPU.

Memory
- Array of memory locations. Can be organized as bits, bytes words or dwords

**Types of Memory**

*RAM*
- Static
- Dynamic

*ROM*
- Masked Rom
- EPROM
- EEPROM
- Flash Memory

**Input/Output**
- Memory mapped versus i/o mapped
Some processors have a separate address space for i/o devices (e.g. 8080, Z80, 80x86), other processors have only a memory address space and expect that i/o devices will be in the same address space as memory. (e.g. 68HC11, 68000)

**Parallel or Serial Interface**

**Serial I/O** -- Data is presented in a bit serial format. Varies with word size, and bit order as well as type of clocking
- Asynchronous
- RS-232
- Synchronous
- USB
- PS/2 keyboard and mouse

**Parallel I/O** -- Data is presented with all data bits at the same time on multiple lines.
- IEE-1284 – Printer port
- SCSI interface

**Interrupt Logic**

Used by high priority i/o devices to break into the CPU’s stream of execution and temporarily cause it to execute other code.

**System Bus** - Used to interface between different parts of the computer system
- Address Bus: Carries address information from the CPU
- Data Bus: Carries instructions and data between the CPU and memory and i/o devices
- Control Bus: Carries control and timing signals between the CPU and the rest of the system

**Microprocessor Operation**

**Machine Cycles**
- Instruction Fetch
- Instruction Decode
- Instruction Execution

**Bus Cycles**
- Memory Read
- Memory Write
- I/O Read
- I/O Write
- Interrupt Acknowledge

**Control Signals from Bus**
- Interrupt
- Non-Maskable Interrupt
- Wait
- Hold (for DMA)
- Multiprocessor synchronization
Development Process

- The assembly language development process uses normal Compile/Link/Debug cycle.
- The assembler generates object modules that are combined by a linker to produce an executable program. Assembler modules can be linked with compiled modules from HLL (e.g. C) to produce what is called a mixed mode program.
- In assembly language, each statement corresponds to a machine instruction.

Statement format:
label: opcode operand1,operand2

- Some statements contain true opcodes, which correspond to machine instructions.
- Some statements contain pseudo-op’s which are instructions to the assembler.
- Instruction operands specify the data, or the location of the data for the instruction to work on.

Data Types and Assembler Data Declaration Pseudo-Ops

Data Sizes: Operands will have a size which is one of the following.

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Size Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BYTE</td>
<td>8 bits</td>
</tr>
<tr>
<td>WORD</td>
<td>16 bits, 2 BYTES</td>
</tr>
<tr>
<td>DWORD</td>
<td>(double word) 32 bits, 2 WORDS or 4 BYTES</td>
</tr>
<tr>
<td>QWORD</td>
<td>(quad word) 64 bits, 4 WORDS, 8 BYTES</td>
</tr>
<tr>
<td>TBYTE</td>
<td>80 bits, 10 BYTES</td>
</tr>
</tbody>
</table>

Character data are arrays of BYTES

Pseudo-Ops

- DB - Define Byte
- DW - Define Word (16 bits)
- DD - Define Doubleword (32 bits)
- DQ - Define Quadword (64 bits)
- DUP - assembler pseudo function to replicate data
- EQU - define symbol

Programmer’s Model of the Machine

Memory Organization:
- 8086 processor has 20 bit address bus, can address 1 MB of memory
- 80286 has 24 bit address bus, can address 16MB of memory
- 80386 and later has 32 bit address bus, can address 4GB of memory

Memory above 1MB boundary can only be accessed in protected mode.

- Original IBM PC had memory organized as array of bytes
- PC-AT had memory organized as array of words
- 80386 and later processors have memory organized as array of dwords

- In all cases, memory is byte-addressable. There are no alignment restrictions, but performance can be increased by using proper alignment.

- In all cases, memory is segmented and all addresses use both a segment and offset. In many cases, the segment is implied in the operation of the instruction. The segment address used by an instruction will always come from one of the cpu’s segment registers.

  Real mode physical address calculation:
  \[ PA = \text{seg} \ll 4 + \text{off} \]

  Interrupt vector table is 1k bytes starting at 0000:0000 (0:0-0:3FFh)
  Reset vector is at FFFF:0