Lecture 1 Notes

Abdulla Nasser

November 10, 2021

Contents

1 Course Labs Materials

- **TASM :** Turbo Assembler/Turbo Debugger, one of the most popular assembler used in the world today. it's used to study the architecture and working of various Microprocessors such as 8085,8086,8051,Pentium Series etc..
- **Emulator 8086 :** 8086 Microprocessor Emulator, also known as EMU8086, is an emulator of the program 8086 microprocessor.
- **Debug program :** External command in DOS used to do assemble for assembly programs and unassemble them. $Assembly \xleftrightarrow{\text{From/To}} Machinecode$. Using this will help you to see/edit memory content and access any memory location.
- **DOS :** Disk operating system that runs from a disk drive. Computer memory space was limited, and when the instructions to control a computer were moved onto a disk drive, such as a floppy disk or internal HDD, it was considered cutting-edge technology.
- **Dos-Box :** Give you DOS environment to run the debug program.

2 Introduction

- Assembly is low level language, Machine code represented with 0's and 1's.
- Basic input/output system, you will deal with when you need I/O. It has some instructions to deal with input and output system.
- We need to utilize our resources to make efficient code.
- Instruction is every statement starts with operation, operation in high level like (+, -, *, /) and in assembly like ADD, and in machine code like 01110010.
- Every processor has specific instructions and it's specific machine code which it only understands.
- The compiler internally call the assembler which take assembly and convert it to machine code.
- You can do same operations you did in high level languages in assembly. so you will find instructions to do arithmetic and logic operations and also for dealing with I/O devices and memory.

3 You need to know your computer

- What you need really to understand well are computer system, memory, CPU and I/O Devices and how they interact together via system bus, which is the main bus.

3.1 Memory

- Memory consists of locations and every one has address. processor needs address to access it in memory. every location is called word and has its size called word size and it is measured by how many bits can I store inside. So 1 MB memory has words each one is 8 bit.

- We represent address in hexadecimal,any number ends with H or starts with 0x is hexadecimal. So 500H is 3 digits ... so it represented in 12 bits (3*4).

- Memory has 2 important things, instruction(operation) and data. Instruction consists of operation and address.

- Load 361 $\rightarrow$ Put value from address 361 (which has the data) inside accumulator(register). 361H is address. Instruction could be 0 address instruction or 1 address or n addresses.

- Check this and search for more about RAM and ROM.

3.2 What happen between memory and CPU?

1. When instruction are in Hard-disk they are idle, after moving them into memory they become active.

2. Memory has running program, program consists of instructions and data. CPU goes to memory for specific address (1st instruction of program execution, goes to the start point which the program loader prepare for)/ memory address register. So it fetch it, instruction will be moved from memory to CPU in the instruction register. Data from memory moved to data bus, then moved to data register/ memory buffer register/ dx to make CPU deal with. . Register is internal memory inside CPU. for fast processing and fast execution.

3. After fetching the instruction, it decode it, to understand opcode part and address part. So it can send signals to circuits responsible for the operations

3.3 CPU

- Every processor has its own sheet that contains all opcodes and it's binary representation. like this and this for 8051
- CU is control unit which initiation signals sent from.
- Address bus holds addresses so it will interact with the memory controlling unit. which response with it's content. System bus consists of three types of wires, called lines, every line holds 1 bit. and it contains sub buses. Check this for advanced explanation. The width of Bus is the number of lines or its size.
- If address is 12 bit, address bus should be 12 bit so every line holds 1 bit. For example but this isn't necessary. memory controlling unit will take these data from the bus and will response for CPU with data wanted which will be put on Data Bus.
- Control Bus Make memory understand the kind of the operation. It hold control signals.
- If $addressbus = 12$ bit so there will be 2^{12} expected place so by knowing this we can know the size of memory and how many places could be used. From data bus I can expect how many words can processor export once.
- There 's type of bus called multiplexed bus. Instead of standalone address bus and data bus, It will create only one bus to hold data or address and the control signal will determine if this is data or address.
- x86 is the basic model for high performance processors. It's Intel family.
- The Logical view of processor in IBM PC book was divided into 2 things. EU: registers dealing with ALU and CU and BIU: registers dealing with memory.

4 Intel 8086 architecture

The interrupt vector table (IVT) is always located in RAM. By default it's located at 0000:0000 at the start of memory, but it's possible to move it using the LIDT instruction. MS-DOS doesn't move the IVT, but Linux might. Either way it will be in RAM somewhere.

Interrupt 0x80 isn't a standard MS-DOS or BIOS interrupt and so normally goes unhandled under MS-DOS. If it's used it all, it's been handled by some third party code (eg. a TSR or maybe a driver) that isn't part of the operating system. Other interrupts might be handled by either MS-DOS (eg. 0x21) or the BIOS (eg. 0x10). In the former case the code for handling the interrupt would be in RAM, while in the later case the code would be in ROM. (Though it's likely the BIOS ROM has been copied to read-only shadow RAM located at the same address of the ROM, as the BIOS code runs much faster this way.)

Under protected mode operating systems like Linux, interrupts are handled exclusively by the operating system.

There's software interrupt and hardware ones. Interrupt handler which is the code that will be executed when that event happen. the interrupt handler exists here as address of the instruction.

Little-endian notation. we divide things as bytes. least byte which is right hand side. the least will be stored in the least address. most significant byte is stored in the greater address. In the registers it will be stored as it is.