Lecture 4 (PC Architecture)

1 Introduction

1. We will be discussing x86 Architecture
2. It is a fairly complex architecture
3. However it is the most common architecture in use right now

2 CPU

1. Components are:
   i. Registers (integer and floating point)
   ii. Memory Management Unit (MMU)
2. Instructions are executed one after the other

3 x86 Architecture

1. The actual processor that will be taken as reference is Intel 8086
2. 4 16-bit registers are present - AX, BX, CX, DX
3. These registers were also divided into halves - ?H, ?L (? = A, B, C, D)
4. 4 more registers were present:
   i. SP = Stack Pointer
   ii. BP = Base Pointer
   iii. SI = Source Index
   iv. DI = Destination Index
5. PC register was also present (gets incremented on every IF (Instruction Fetch), if no branches or jumps)
6. FLAGS register is also present (cannot be modified directly) - flags here are set when some arithmetic or other errors happen
7. AT&T syntax is going to be used for x86
8. 4 segments are also present:
   i. CS - Code Segment
   ii. DS - Data Segment
   iii. SS - Stack Segment

   iv. ES (for anything else)
9. This suggests that more than 2^{16} memory locations are not possible but it is possible to read until 2^{20} ($segment \times 16 + PC$)

Backward compatibility is provided till date, so an OS first boots in 16-bit mode and then it switches to 32-bit (and then to 64-bit) mode as required

4 Extension from 16 to 32 bits

1. Each register has been renamed as E??
2. Old names can still be used to refer the lower 16 bits
3. Instructions have been modified to work with 32-bit registers
4. New prefixes (for each instruction) have been created to work with 16 bits (0x66 for data and 0x67 for address)
5. .code32 directive is used to inform assembler that the instructions after this are in 32-bit by default

5 AT&T Syntax

```
movl %eax, %edx // edx = eax
movl $0x123, %edx // edx = 0x123
movl 0x123, %edx // edx = *(int *) 0x123
movl (%ebx), %edx // edx = *(int32 *)ebx
movl (%ebx), %edx // edx = *(int32 *)(ebx + 4)
```