# x86 Architecture

Basic Structure

Assembly language programming
By xorpd

xorpd.net

# Objectives

- You will understand how programs are stored in memory, and how they are executed by the processor.
- You will learn about the x86 32-bit registers.

# Storing programs in memory

- The program to be run by the processor is written in memory (In RAM).
  - The memory is “made of” bits.
  - A **byte** is a set of 8 consecutive bits.
  - A byte is the basic information unit in x86 processors.
  - Usually a byte will be represented as two hexadecimal digits.
  - Hexadecimal digit (represents 4 bits) is sometimes called a **nibble**.
- Example for a simple program (represented in base 16):
  - 89 C1 01 C9 01 C1
  - Interpretation:
    - 89 C1 : mov ecx,eax
    - 01 C9 : add ecx,ecx
    - 01 C1 : add ecx,eax

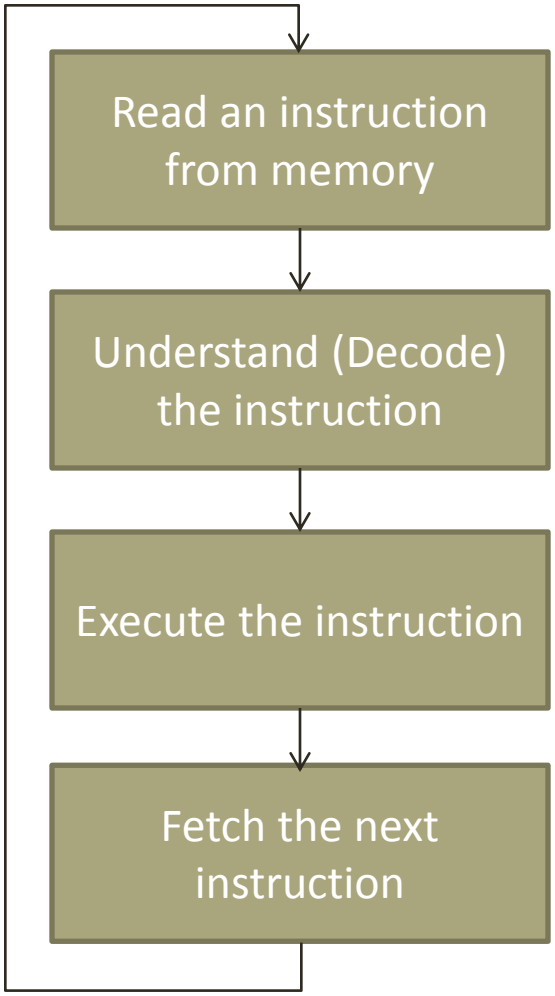
# Storing programs in memory

- The program to be run by the processor is written in memory (In RAM).
  - The memory is “made of” bits.
  - A **byte** is a set of 8 consecutive bits.
  - A byte is the basic information unit in x86 processors.
  - Usually a byte will be represented as two hexadecimal digits.
  - Hexadecimal digit (represents 4 bits) is sometimes called a **nibble**.
- Example for a simple program (represented in base 16):
  - 89 C1 01 C9 01 C1
  - Interpretation:
    - 89 C1 : mov ecx,eax
    - 01 C9 : add ecx,ecx
    - 01 C1 : add ecx,eax

# x86 Instructions

- An Instruction in the x86 architecture represents some simple task.
  - Examples: Addition, subtraction, moving data etc.
- Every instruction is encoded as one or more bytes.
  - Instructions come in various sizes. There are very short instructions (one byte), and very long instructions (sometimes even 10 bytes or more).
  - The numeric representation of an instruction is also called **opcode** (**Op**eration **code**).
- The processor can only understand the numeric representation of the instructions.
- You don’t have to remember the numeric representation of the instructions. There is a textual representation for us humans.
  - Eventually though, you might remember some of the numeric representations of the instructions.

# Processor's operation cycle



# The registers

- x86 processors have some very efficient internal places to store data. These are called **registers**.
  - Built inside the processor. (Not on the RAM).
  - Very efficient.
  - Very scarce – There are only a few.
  - Represent the internal state of the processor.
  - As usual, made of bits.
- We will learn about the names of existent registers. Only later we will learn about the things we can do with them.
- Don’t worry if in the beginning you don’t remember the names of all the registers.

# Basic registers

- Basic registers, each is made of **32 bits**:
  - eax – Accumulator.
  - ebx – Base index.
  - ecx – Counter.
  - edx – Data register.
- In early x86 processors every register had a specific job for specific operation.
  - In recent processors (Mostly in protected and Long modes), registers became more general purpose.
  - The old roles of the registers can help you remember their names.
- The “e” stands for extended.

# Register extensions

- The 32-bit registers are extensions of the old 16 bit registers.
  - Due to backwards compatibility.

<table>
<tr><td colspan="3">eax (32 bit)</td></tr>
<tr><td></td><td colspan="2">ax (16 bit)</td></tr>
<tr><td></td><td>ah (8 bit)</td><td>al (8 bit)</td></tr>
</table>

- ax is just another name for the lowest 16 bits of eax.
  - al, ah are just another names for bits 0:7 and bits 8:15 of eax respectively.
  - h stands for high, l stands for low.
- Example:
  - If eax contains 0xABCD1234, then:
    - ax contains 0x1234
    - ah contains 0x12. al contains 0x34.

# Register extensions (Cont.)

| eax (32 bit) | | |
|---|---|---|
| | ax (16 bit) | |
| | ah (8 bit) | al (8 bit) |

| ebx (32 bit) | | |
|---|---|---|
| | bx (16 bit) | |
| | bh (8 bit) | bl (8 bit) |

| ecx (32 bit) | | |
|---|---|---|
| | cx (16 bit) | |
| | ch (8 bit) | cl (8 bit) |

| edx (32 bit) | | |
|---|---|---|
| | dx (16 bit) | |
| | dh (8 bit) | dl (8 bit) |

# 64 bit extensions (Long mode)

- The 32-bit registers were later extended again, to 64 bits:

<table>
<tr><td colspan="4">rax (64 bits)</td></tr>
<tr><td colspan="2"></td><td colspan="2">eax (32 bits)</td></tr>
<tr><td colspan="3"></td><td colspan="1">ax (16 bits)</td></tr>
<tr><td colspan="3"></td><td>ah (8)</td><td>al (8)</td></tr>
</table>

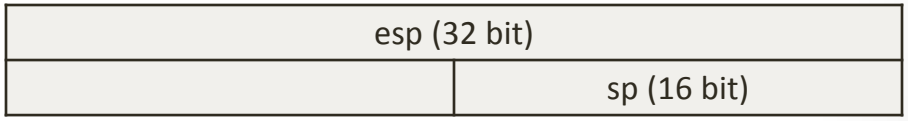
- We are mostly going to deal with 32-bits registers in this course.
  - Almost every 32-bit register has a 64-bit equivalent.

# More registers

- Index registers:
  - esi – Source Index register.
  - edi – Destination Index register.


- Instruction Pointer:
  - eip.


- Flags register.
- Stack pointers:
  - esp – Stack Pointer.
  - ebp – Base Pointer.




- There are even more registers.

# Summary

- **Programs** are just a bunch of bytes.
- The processor can read bytes and interpret those as a program.
- There are places to store data inside the processor called **registers**. They represent the inner state of the processor.
  - Registers (In this course) are of size 32 bits.
  - Big registers are consisted of smaller registers, due to backwards compatibility.

- Don’t worry if you don’t remember the names of the registers. We are going to learn more about them in the following lectures.