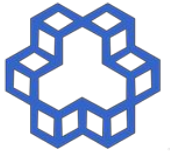# Introduction to 8086 Assembly

Lecture 1

Behrooz Nasihatkon
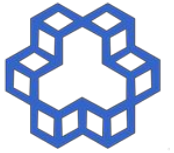
Introduction to 8086 Assembly Language

- 3 credits
- Saturday, Wednesday 15:30-17:30 AM
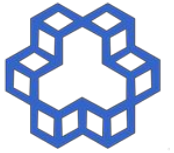- Instructor: Behrooz Nasihatkon
- Email: nasihatkon@kntu.ac.ir

Roll call

What is considered cheating?
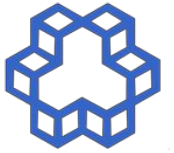
Special needs

COVID-19

Auditing the course

Recording the lectures
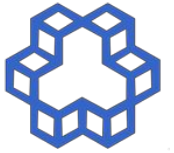
Eating in class
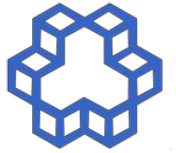
How to get help?
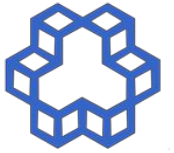
Asking questions!

Grading

- Homework Assignments
- Project(s)
- Midterm Exam(s)
- Final Exam
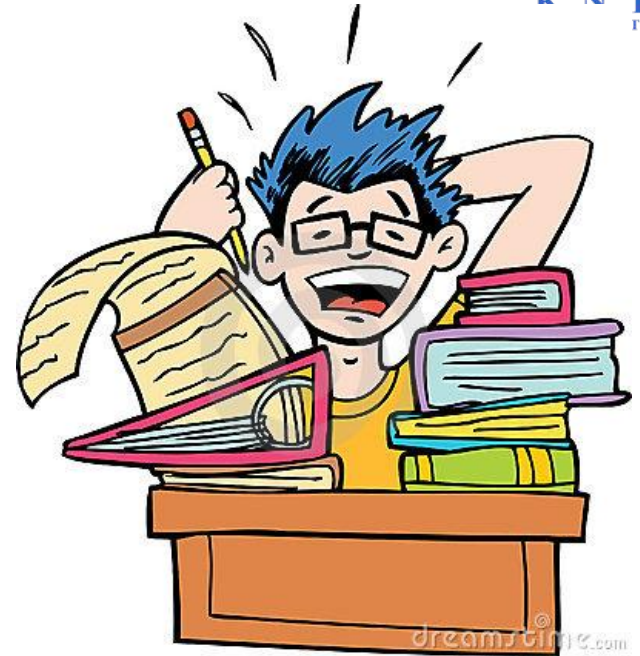
The exams

- Tools you need
    a. A webcam
    b. A microphone
    c. A screen recorder (preferably OBS-studio)
    d. An md5 checksum generator

- For each question
    a. Record a video of yourself solving the question
    b. Upload the solution
    c. Record a video explaining your solution
    d. Upload the md5sums of the videos

How to give feedback?
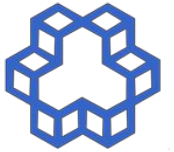
Anonymous form:
https://goo.gl/zPxBAS

Join the Telegram Channel

https://t.me/asmkntus00

Course Website

- https://wp.kntu.ac.ir/nasihatkon/teaching/asm/f2020/index.html
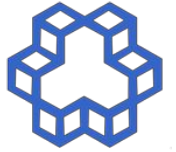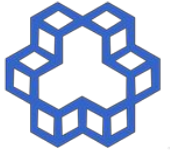
Resources

- Carter, Paul A. **PC Assembly Language**, 2007
  - http://pacman128.github.io/pcasm/
- **NASM tutorial**
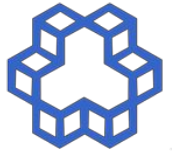 - http://cs.lmu.edu/~ray/notes/nasmtutorial/
- **TutorialsPoint**
  - https://www.tutorialspoint.com/assembly_programming
- **GOOGLE!**

Further study:

- Hyde, Randall. **The art of assembly language**. No Starch Press, 2010.
  - **Linux:** http://www.plantation-productions.com/Webster/www.artofasm.com/Linux
  - **Windows:** http://www.plantation-productions.com/Webster/www.artofasm.com/Windows/
- Blum, Richard. **Professional assembly language**. John Wiley & Sons, 2007.

What is Assembly language?
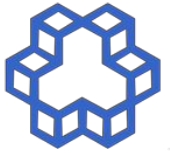
What is Assembly language?
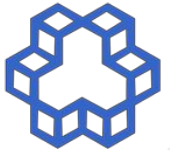
What is Assembly language?


```
        call read_int
        mov ecx, eax

        call read_int

        mov ebx, 0
l1:
        add ebx, eax

        loop l1


        mov eax, ebx
        call print_int
        call print_nl
```

What is Assembly language?
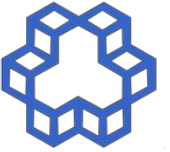
How many assembly languages are there?

How many assembly languages are there?

Why assembly?
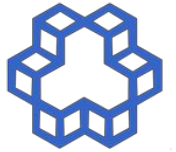
Why assembly?

- Going low-level!
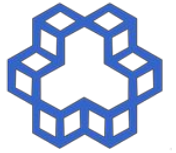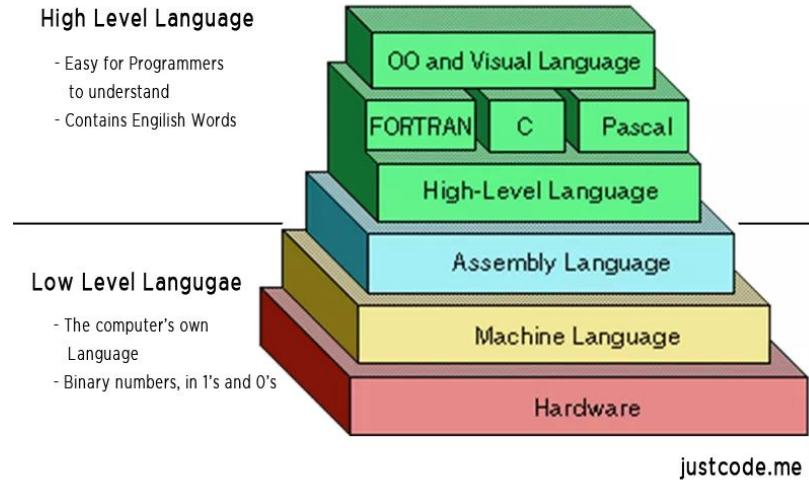- **Getting insight**
  - **How programming languages are implemented (code, variables, arrays, functions, etc.)!**
  - **How compilers work**
- Writing efficient programs (?)
- System programming
- Writing device drivers
- Interfacing with high-level languages like C
- Reverse engineering
- New CPU features

x86 & x86-64 Assembly

AT&T vs Intel Syntax

https://en.wikipedia.org/wiki/X86_assembly_language#Syntax

```
movq    %fs:40, %rax
movq    %rax, -8(%rbp)
xorl    %eax, %eax
leaq    -16(%rbp), %rax
movq    %rax, %rsi
movl    $.LC0, %edi
movl    $0, %eax
call    __isoc99_scanf
movl    -16(%rbp), %eax
addl    %eax, %eax
leal    3(%rax), %edx
movl    -16(%rbp), %eax
imull   %edx, %eax
movl    %eax, -12(%rbp)
```
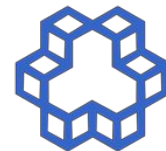
```
sub     rsp, 16
mov     rax, QWORD PTR fs:40
mov     QWORD PTR [rbp-8], rax
xor     eax, eax
lea     rax, [rbp-16]
mov     rsi, rax
mov     edi, OFFSET FLAT:.LC0
mov     eax, 0
call    __isoc99_scanf
mov     eax, DWORD PTR [rbp-16]
add     eax, eax
lea     edx, [rax+3]
mov     eax, DWORD PTR [rbp-16]
imul    eax, edx
```

What is an **Assembler**?
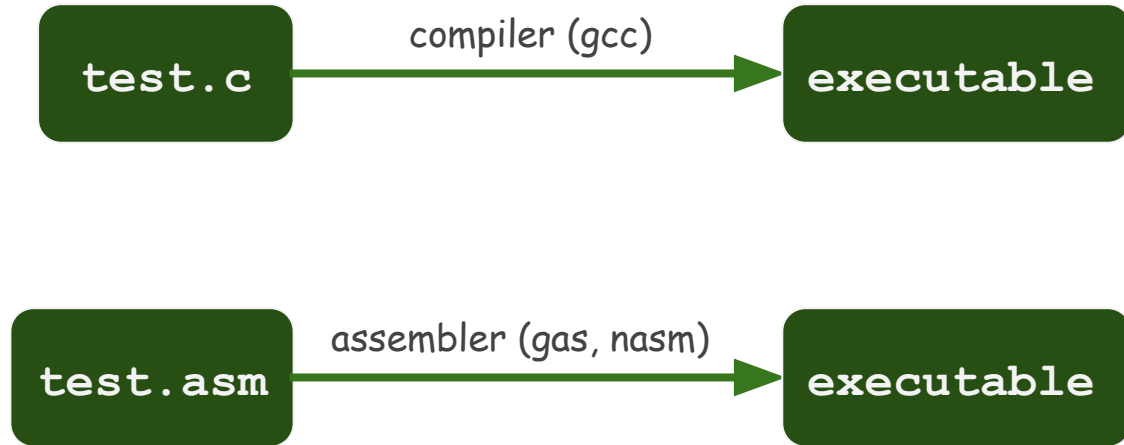
```
test.c  ──── compiler (gcc) ────▶  executable
```

```
test.asm  ──── assembler (gas, nasm) ────▶  executable
```
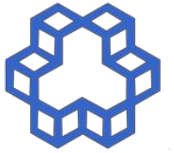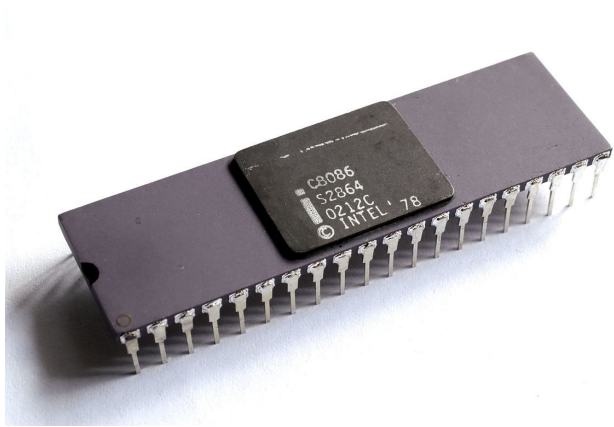
Major Assemblers

- Microsoft Assembler (MASM)
- GNU Assembler (GAS)
- Flat Assembler (FASM)
- Turbo Assembler (TASM)
- Netwide Assembler (NASM)

Major Assemblers

- Microsoft Assembler (MASM)
- **GNU Assembler (GAS)**
- Flat Assembler (FASM)
- Turbo Assembler (TASM)
- **Netwide Assembler (NASM)**

Backward compatibility

- Look at
  - https://en.wikipedia.org/wiki/X86

Our platform

- **Hardware**: 80x86 processor (**32**, 64 bit)

Our platform

- **Hardware**: 80x86 processor (**32**, 64 bit)
- **OS:**
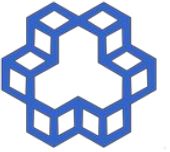
Our platform

- **Hardware**: 80x86 processor (**32**, 64 bit)
- **OS**:
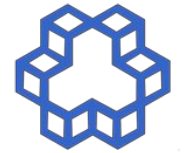
Our platform

- **Hardware**: 80x86 processor (**32**, 64 bit)
- **OS**:

Our platform

- **Hardware**: 80x86 processor (32, 64 bit)
- **OS**: Linux

Our platform
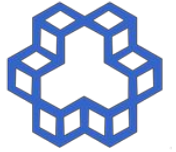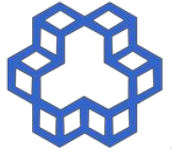
- **Hardware**: 80x86 processor (**32**, 64 bit)
- **OS**: Linux
- **Assembler**: Netwide Assembler (NASM)
  - + GNU Assembler (GAS)
- **C Compiler**: GNU C Compiler (GCC)
- **Linker**: GNU Linker (LD)

How does an assembly code look like?

Write a C program named **test.c**.

Compile it to x86 assembly language, the **AT&T syntax**

>>> gcc -S -o att.s test.c

Now compile to the **Intel syntax**:

>>> gcc -S -masm=intel -o intel.s test.c

Compare the two assembly syntaxes (output files att.s and intel.s)