Mansoura University
Faculty of Computers and Information
Department of Computer Science
First Semester: 2020-2021

[CS214P] Assembly Language: An Introduction

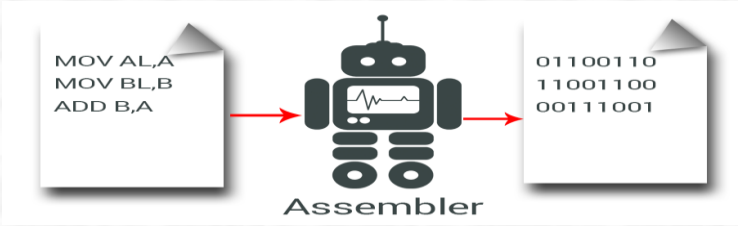
Grade: Third Year (Computer Science)

Sara El-Metwally, Ph.D.

Faculty of Computers and Information,

Mansoura University,

Egypt.



Computer Science Department
Faculty of Computers and Information
Mansoura University

Assembly Language: An Introduction

"Basic Features of PC Hardware"

Sara El-Metwally, Ph.D.
Faculty of Computers and Information,
Mansoura University, Egypt.

Email: sarah_almetwally4@mans.edu.eg
sara.elmetwally.2007@gmail.com

Course outlines

- **Course Meeting Time:**

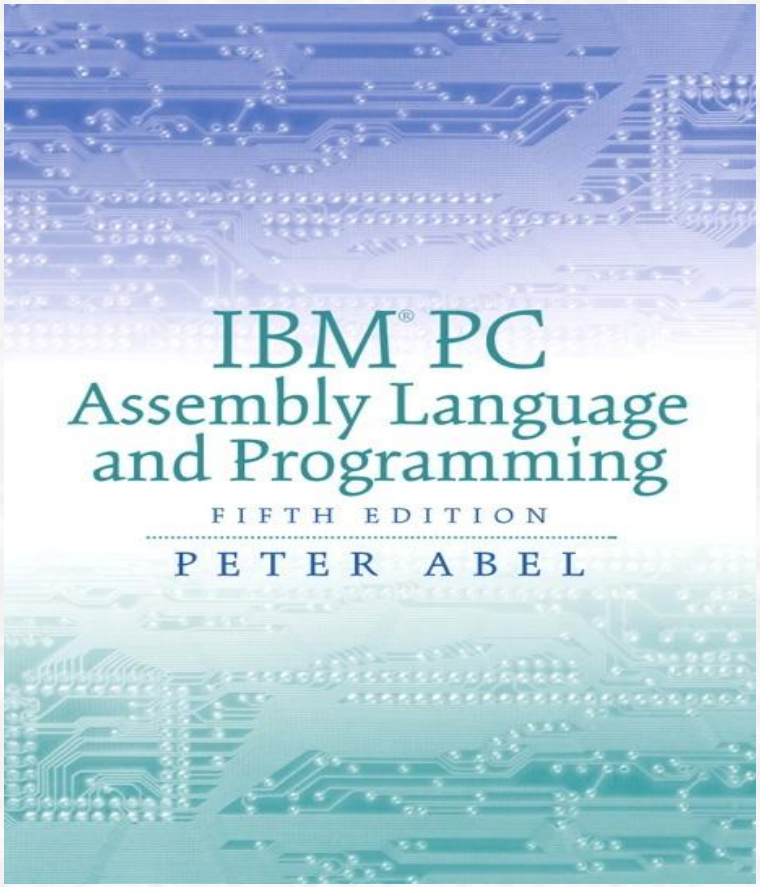
  Sunday, 12.20 pm – 2:00 pm

- **Grading :**

Activities	Percentages
Attendance (Lectures + Sections)	5%
Practical exam	15%
Midterm	10%
Oral	10%
Final	60%

Course outlines

- **Course Text Book:**



Course outlines

- **Course Labs " materials ":**
  - Assignments and solving some exercises.
  - DosBox.
  - Debug Program.
  - Emulator 8086.
  - AE-TASM.
- **Course TA's:**
  - Eng. Ghada Shafiq

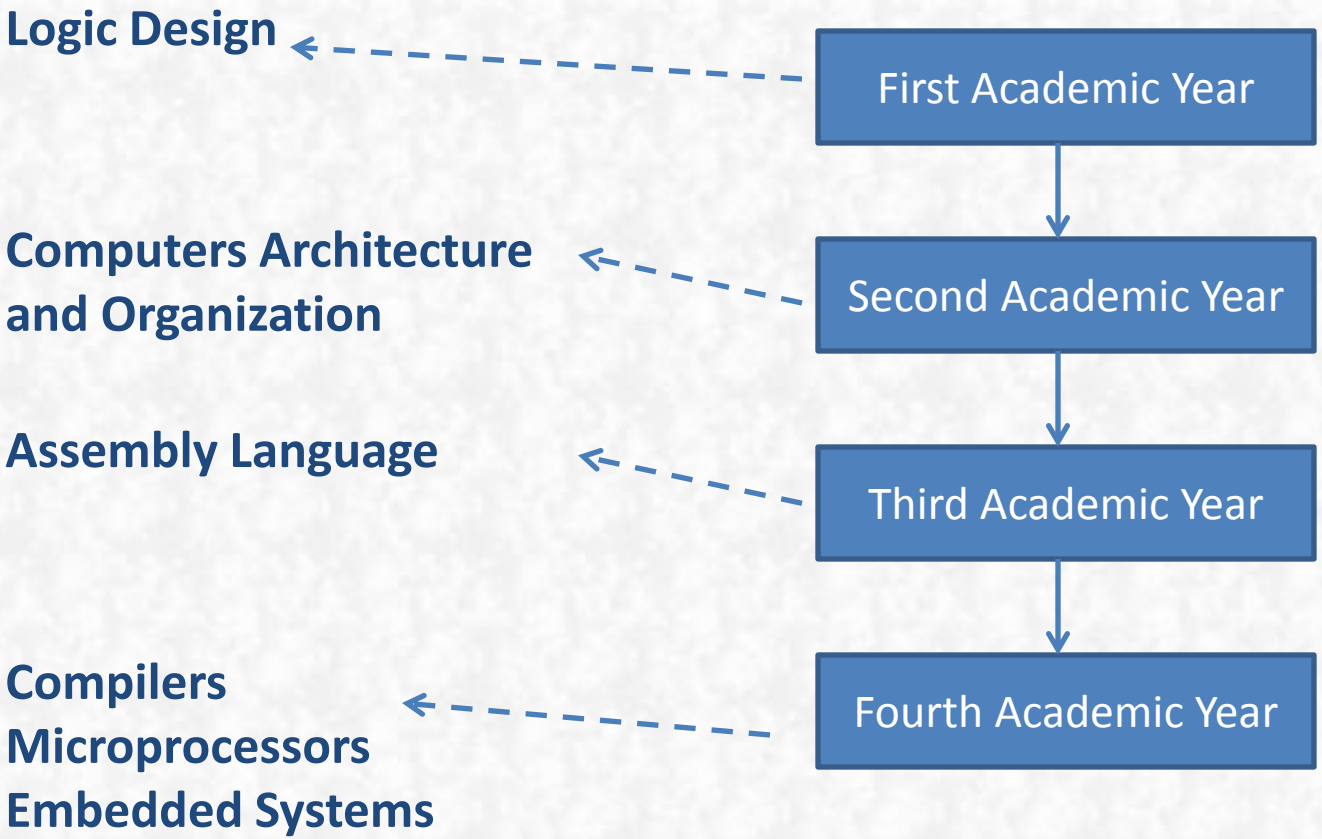
Course Objectives

- Understand the HW of the personal computer.
- Understand machine-language code and hexa-decimal format.
- Understand the steps involved in assembling, linking, and executing a program.
- Write programs in assembly language to handle the keyboard and screen, perform arithmetic, convert between ASCII and binary formats, perform table searches and sorts, and handle disk I/O.

Course Objectives

- Trace machine execution as an aid in program debugging.
- Write your own macro instructions to facilitate faster coding.
- Linking separately assembled programs into one executable program.

Course Requirements



Why Assembly? (Knowledge)

- Shows how program interface with operating system, processor and BIOS.
- Show how data is represented and stored in memory and on external devices.
- Show how processors access and execute instructions and how instructions access and process data.
- Show how a program access external devices.

Why Assembly in 2019?

- Programs written in assembly requires less memory and execution time.
- Give programmers the flexibility to perform low level machine operations and highly technical tasks.
- While most software written in high-level programming languages, a common practice is to recode in assembly those sections that are time-critical.

Why Assembly in 2019?

```
__m128i inline sse_next_16_bit_rc( char * s){
    //__m128i m0=_mm_set_epi8(s[15],  s[14],  s[13],  s[12],  s[11],  s[10
    __m128i m0=_mm_loadu_si128((const __m128i *)s);
    //__m128i m0=_mm_set_epi8(s[0],  s[1],  s[2],  s[3],  s[4],  s[5],  s[
    __m128i I_a, I_c, I_g, I_t;
```

Turtle: Identifying frequent *k* -mers with cache-efficient algorithms FREE

Rajat Shuvro Roy ✉, Debashish Bhattacharya, Alexander Schliep ✉

Bioinformatics, Volume 30, Issue 14, 15 July 2014, Pages 1950–1957,
https://doi.org/10.1093/bioinformatics/btu132
Published: 10 March 2014 **Article history** ▾

```
    m0=_mm_or_si128(comp_a, comp_c);
    m0=_mm_or_si128(m0, comp_t);

    return m0;
    }
```

Modern computers support SSE (Patterson and Hennessey, 1998) instructions that operate on 128-bit registers and can perform arithmetic/logic operations in parallel on multiple variables. We used Streaming SIMD Extensions (SSE) instructions for speeding up bit encoding of *k* -mers.

Why Assembly in 2019?

https://cs.stackexchange.com/questions/13287/why-do-we-need-assembly-language

"Why am I studying about assemblers in my computer engineering class?"

> Though it's true, you probably won't find yourself writing your next customer's app in assembly, there is still much to gain from learning assembly.
>
> Today, assembly language is used primarily for direct hardware manipulation, access to specialized processor instructions, or to address critical performance issues. Typical uses are device drivers, low-level embedded systems, and real-time systems.
>
> Assembly language is as close to the processor as you can get as a programmer so a well designed algorithm is blazing -- assembly is great for speed optimization. It's all about performance and efficiency. Assembly language gives you complete control over the system's resources. Much like an assembly line, you write code to push single values into registers, deal with memory addresses directly to retrieve values or pointers. (source: codeproject.com)

share cite improve this answer

edited Jul 15 '13 at 17:33

answered Jul 15 '13 at 15:35

TylerAndFriends

343 ● 1 ● 8

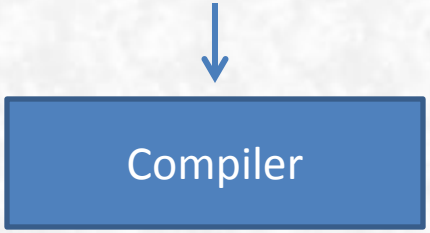
Levels of Programming

- **Machine language**: instructions in a sequence of ones and zeros that the processor executes one at a time.
- **Low-level assembly language**: instructions in a symbolic format designed for a specific processor and have one to one machine code.
- **High-level language**: designed to eliminate the technicality of low level hardware details and generate many low-level instructions.

Levels of Programming

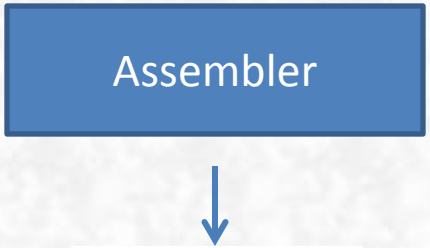
```
unsigned int  bit_seq_i;
unsigned char *bit_seq_buff,
bit seq i=16;
```

High-level language




```
mov ax, data
mov ds, ax
mov es, ax
mov cx, 10
mov dx, 0
```

Low-level assembly language



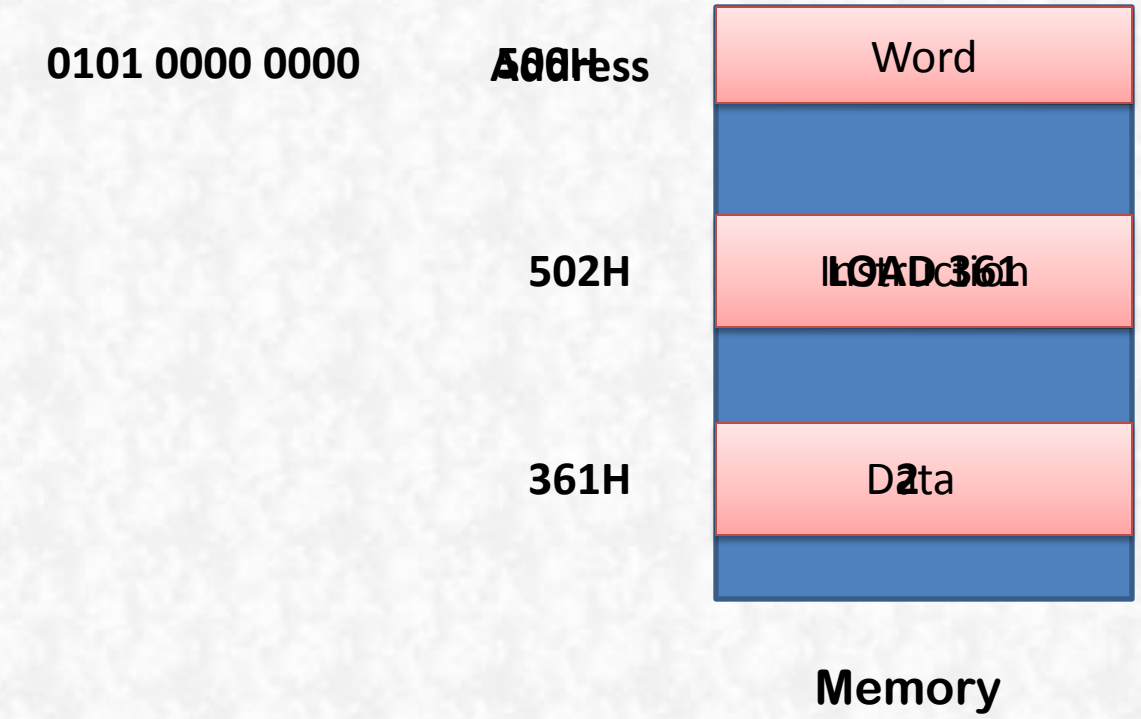

```
0001100001000010001
0010000010000111100
1001001001100001001
```

Machine language

What is Assembly Language?

- **Assembly language** is a low-level programming language for a computer or other programmable device specific to a particular computer architecture.
- Each personal computer has a microprocessor that manages the computer's arithmetical, logical, and control activities.
- Each family of processors has its own set of instructions for handling various operations such as getting input from keyboard, displaying information on screen etc.

To Start:
You need to know your Computer ?



To Start:
You need to know your Computer ?

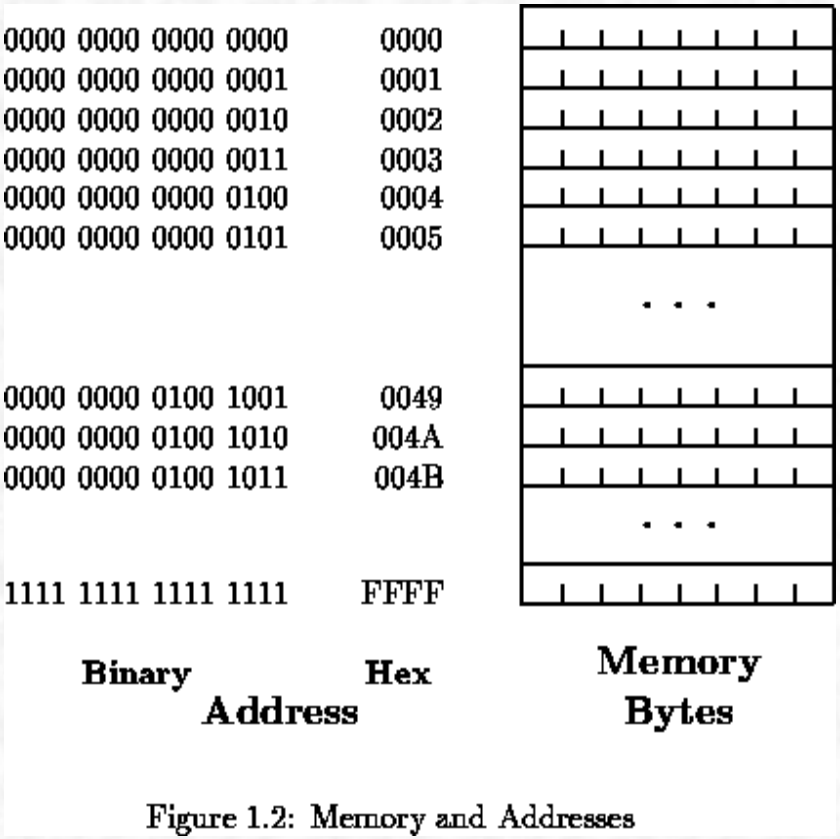
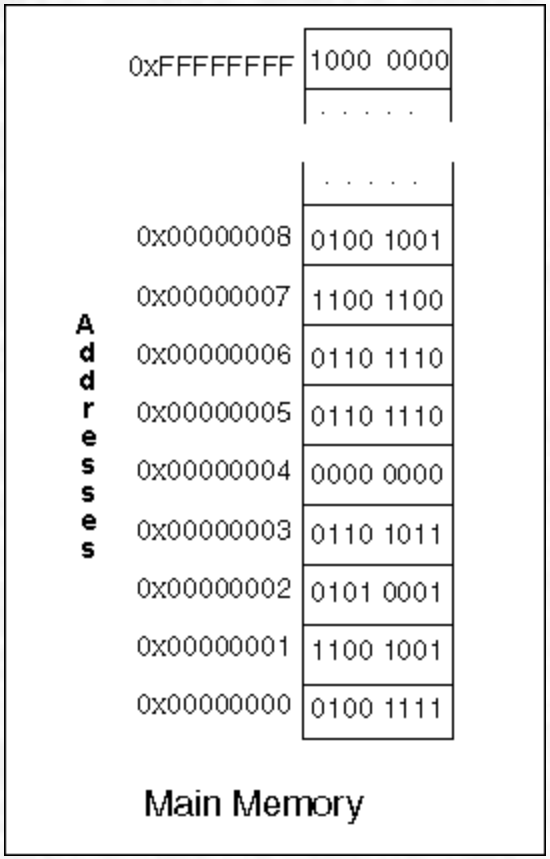
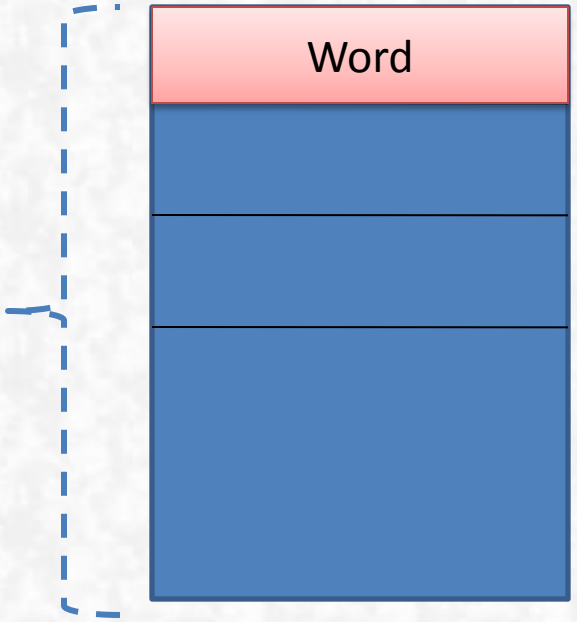


Figure 1.2: Memory and Addresses



To Start:
You need to know your Computer ?



Memory
(1MB)

Si

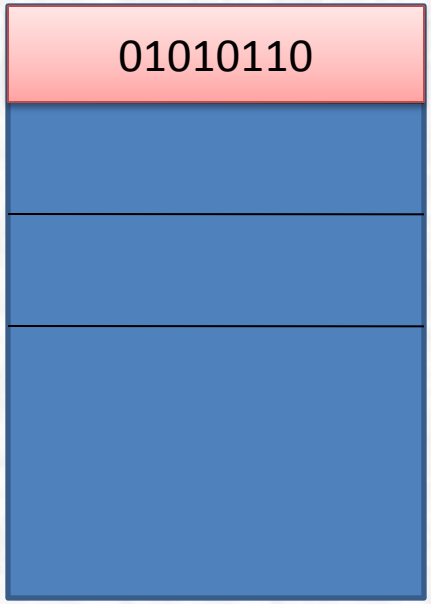
$$memory\ size\ in\ words = \frac{memory\ size\ in\ bits}{word\ size}$$

$$memory\ size\ in\ words = \frac{2^{10} \times 2^{10} \times 2^{3}}{2^{3}} = 2^{20}\ words$$

$$\#\ bits\ for\ address = 2^{20}\ words = 20\ bits$$

Memory size in words

To Start:
You need to know your Computer ?



Memory
(1MB)

0001 0001 0010 0010 1111

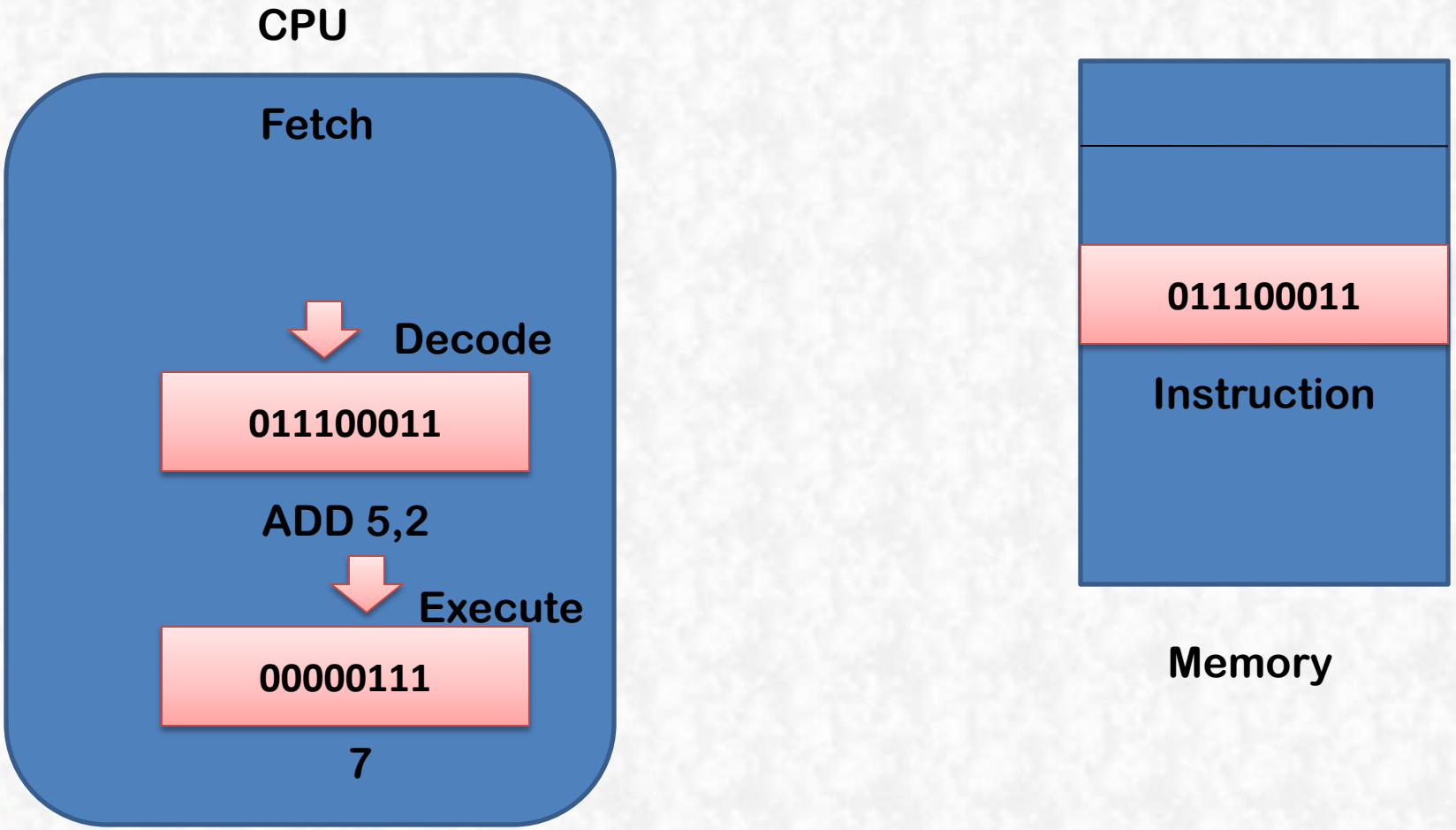
0x1122F

$$memory\ size\ in\ words = \frac{memory\ size\ in\ bits}{word\ size}$$

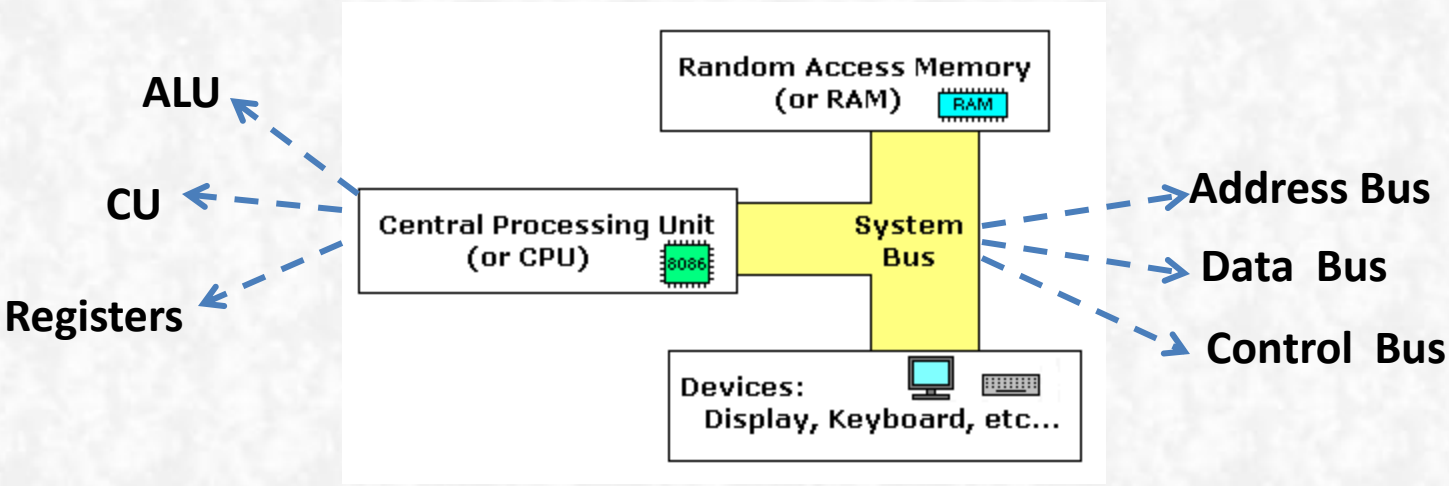
$$memory\ size\ in\ words = \frac{2^{10} \times 2^{10} \times 2^{3}}{2^{3}} = 2^{20}\ words$$

$$\#\ bits\ for\ address = 2^{20}\ words = 20\ bits$$

To Start:
You need to know your Computer ?



To Start:
You need to know your Computer ?



Intel 8086 (x86 architecture)

- 16-bit registers.
- 16-bit data bus.
- 1 MB internal memory.
- Word size = 1 byte.

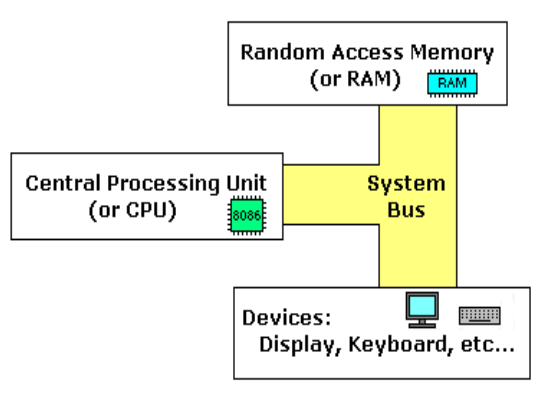
How many bits for address bus?

20 bits

How many words can be fetched from a memory at a time?

2 words

Intel 8086 (x86 architecture)



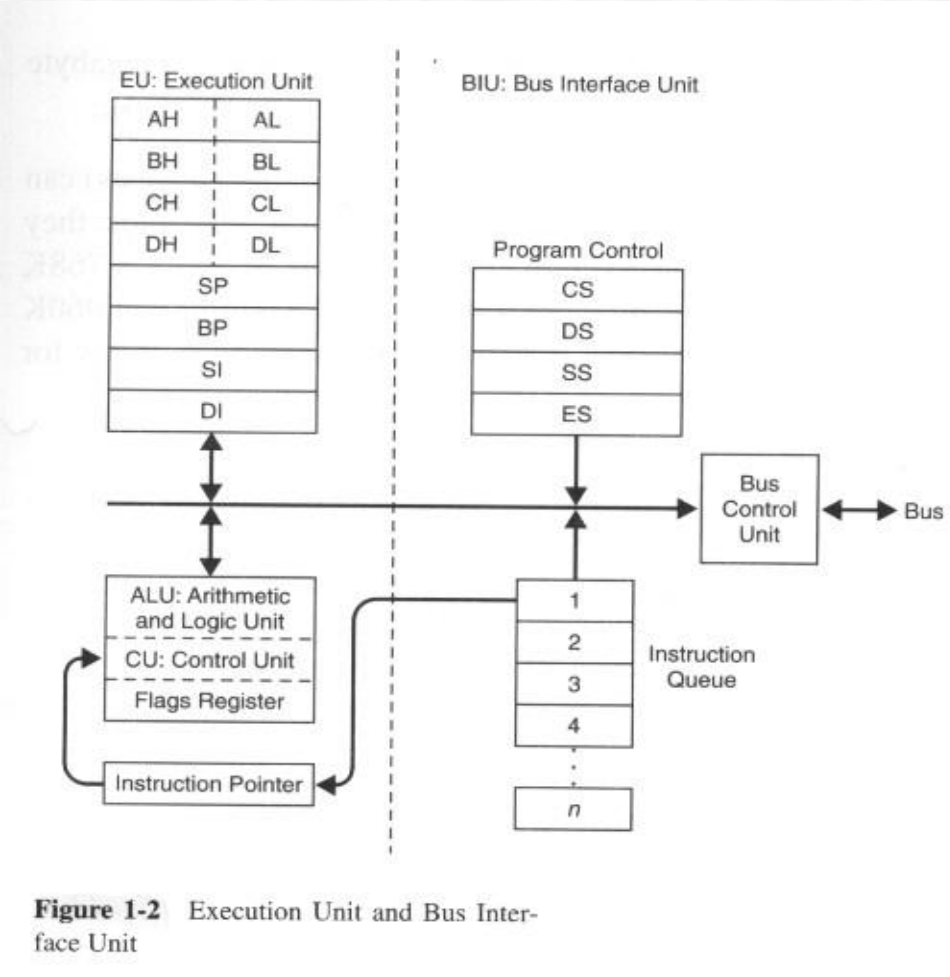


Figure 1-2 Execution Unit and Bus Interface Unit

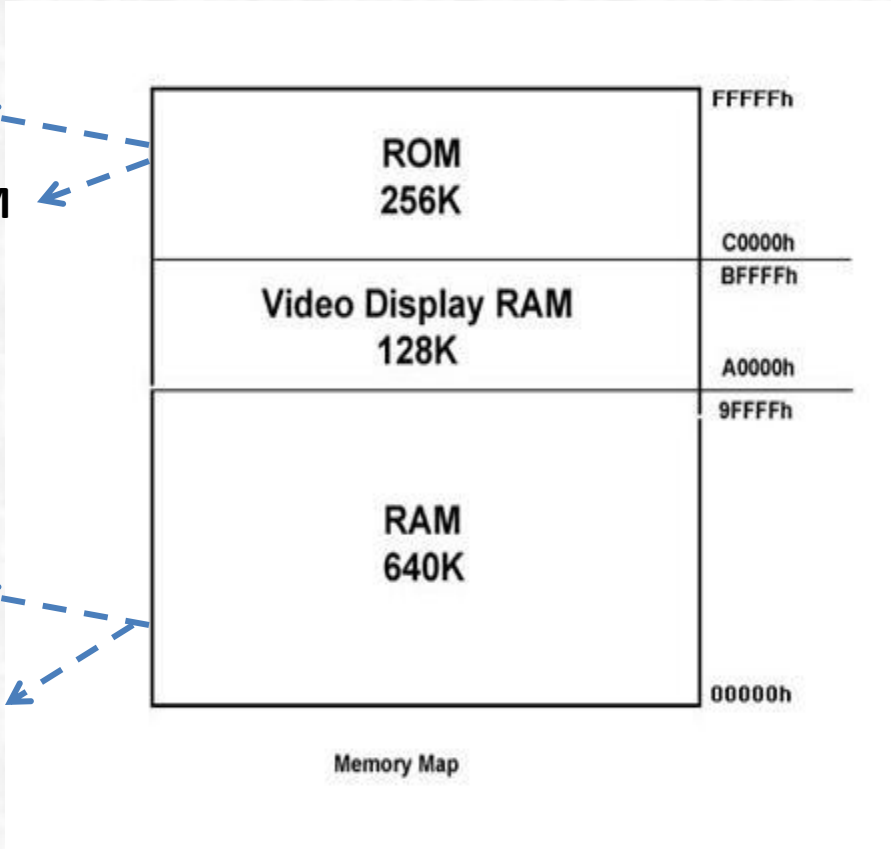
Intel 8086 (x86 architecture)

64K base system ROM

192K memory expansion area ROM

BIOS data area

Interrupt Vector Table



Addressing Data in Memory

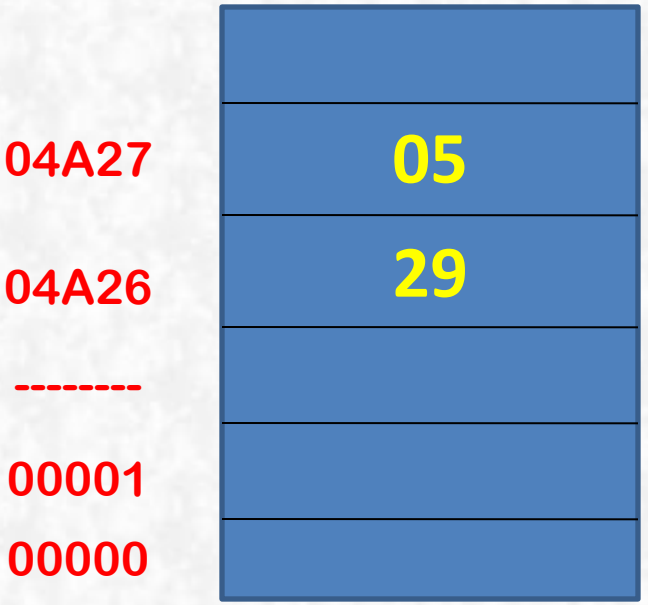
the Intel x86 processors use little-endian.

We need to store 0529H in memory

How many words we need?

2 words

Register
(Queue)



Memory
(Stack)

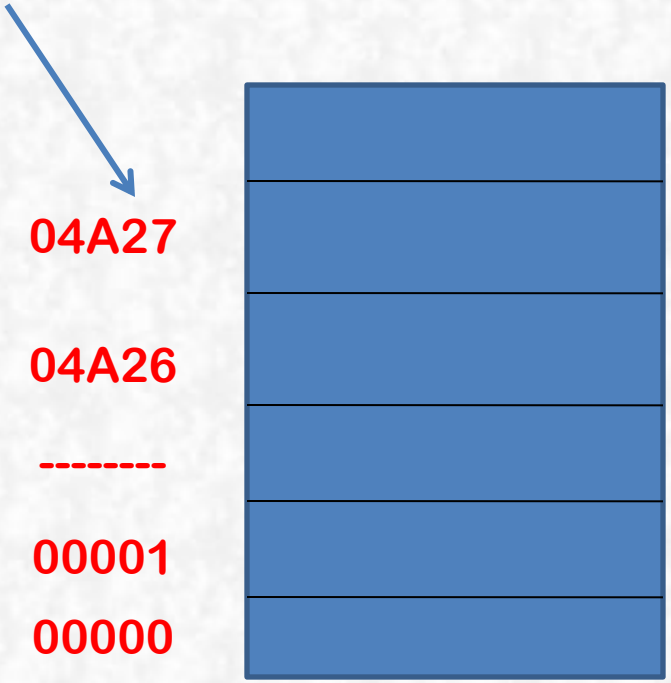
Addressing Data in Memory

- **An absolute address: 20-bit value that directly references a specific location in memory.**

- **A segment : offset address, combines the starting address of a segment with an offset value.**

Addressing Data in Memory

An absolute address



Memory
(Stack)

Addressing Data in Memory