東北大學
Northeastern University

汇编语言程序设计

主讲： 刘松冉

单位： 东北大学计算机学院
智慧系统国际联合实验室

联系方式： liusongran@cse.neu.edu.cn

个人主页： http://faculty.neu.edu.cn/liusongran
https://liusongran.github.io/

第一、二章 内容回顾

第一章 概述

一. 计算机语言

二. 为什么要学习汇编语言

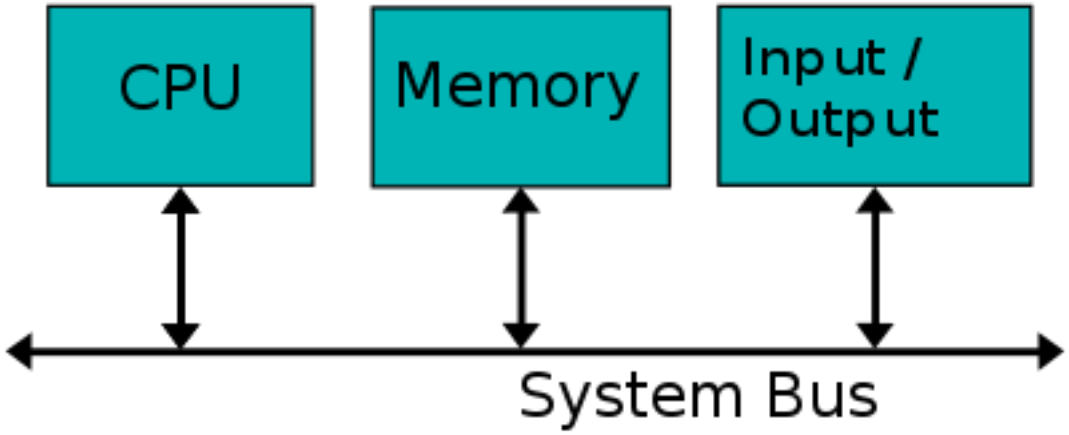
三. 计算机系统

三. 计算机系统

计算机系统由硬件子系统和软件子系统组成。

- **硬件子系统**：组成计算机系统的所有电子的、机械的、光学的和磁性的元部件
  - 中央处理器CPU
  - 存储装置
  - 外围设备
  - 各级总线

计算机程序在？执行
计算机程序在？存储
程序的执行过程是？和？的交互过程



[1] https://sites.google.com/site/sunderlandske2016/week-4---how-computers-work/1-basic-computer-architecture

第二章 计算机运算基础

一. 进位计数制 → 二进制

二. 数制之间的转换

三. 二进制编码

四. 带符号数的机内表示

五. 二进制运算

第三章 微型计算机的结构

一. 微处理器的结构(8086/8088)

二. 存储器(组织与结构)

三. 寻址方式

四. 指令系统(概括)

第三章 微型计算机的结构

一. 微处理器的结构(8086/8088)

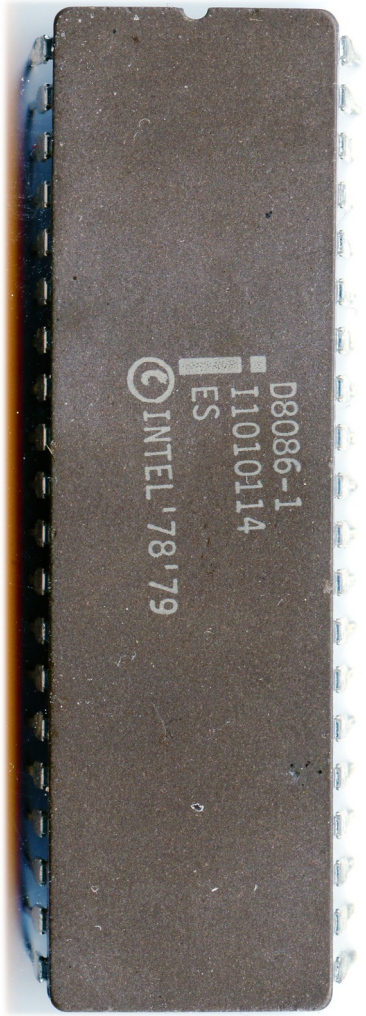
1. 8086/8088 CPU的结构
2. 8086/8088的寄存器组织

一. 微处理器的结构(8086/8088)

1. 8086/8088 CPU的结构

8086 基本信息：

- 1978年6月8日
- 主频：4.77，8和10MHz
- 晶体管：29000个
- 最小特征尺寸：3微米
- 40个引脚（pin）
- 总线宽度：16bit数据总线，20bit地址总线
- 能寻址1M的存储单元和64K的I/O端口



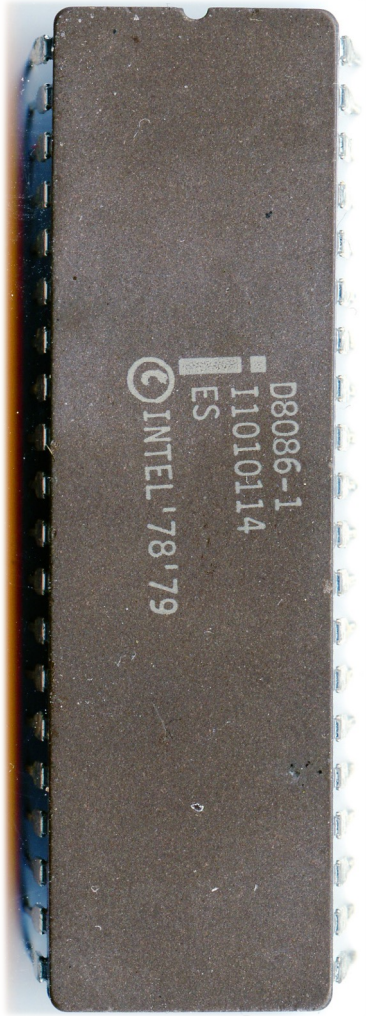
[1] https://www.engineering-sample.com/gallery/Intel/007%208086/D8086-1%20ES/
[2] https://www.cpushack.com/life-cycle-of-cpu.html

一. 微处理器的结构(8086/8088)

1. 8086/8088 CPU的结构

8086 基本信息：

- 1978年6月8日
- 主频：4.77，8和10MHz
- 晶体管：29000个
- 最小特征尺寸：3微米
- 40个引脚（pin）
- 总线宽度：16bit数据总线，20bit地址总线
- 能寻址1M的存储单元和64K的I/O端口



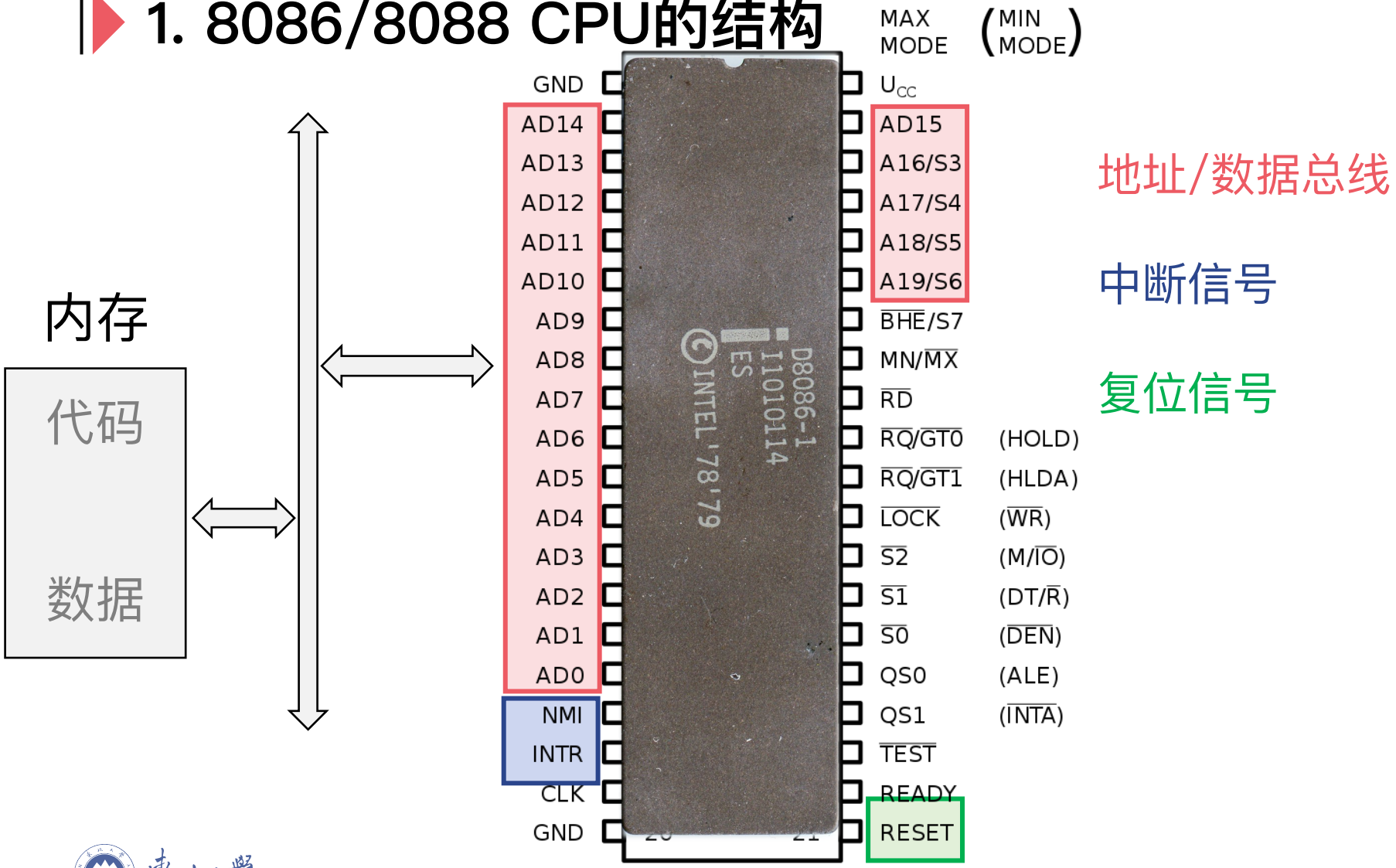
i7-8086K 基本信息：

- 2018年6月8日
- 主频：4，5GHz
- 晶体管：~30亿
- 最小特征尺寸：14纳米
- LGA 1151 v2
- 总线宽度：64bit总线

[1] https://www.engineering-sample.com/gallery/Intel/007%208086/D8086-1%20ES/
[2] https://www.cpushack.com/life-cycle-of-cpu.html

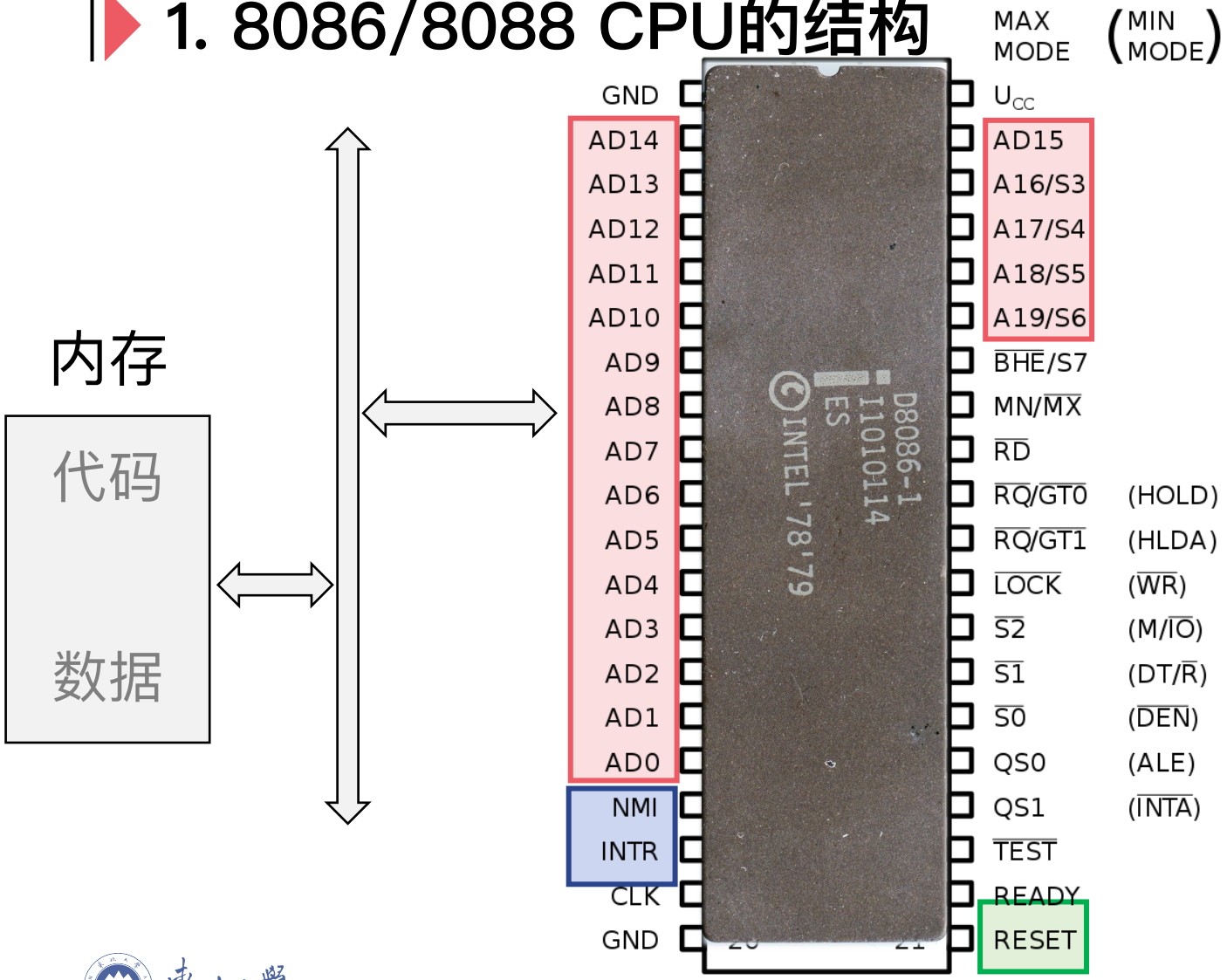
一. 微处理器的结构(8086/8088)

1. 8086/8088 CPU的结构



一. 微处理器的结构(8086/8088)

1. 8086/8088 CPU的结构




```
MOV     BL, 06H
MOV     CL, 05H
ADD     BL, CL
```

GCC Assembler [1]

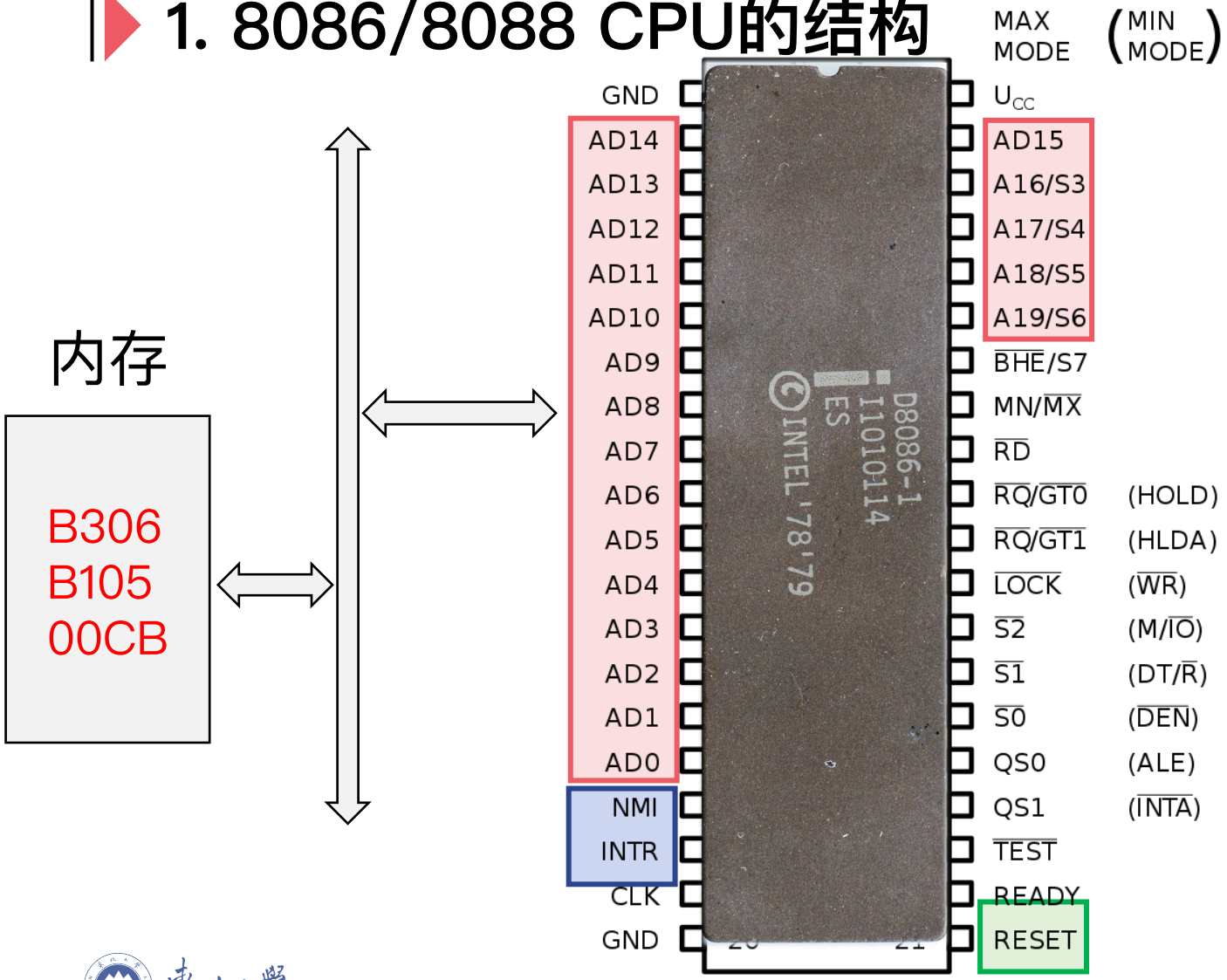
```
B306
B105
00CB
```

机器码
Raw Hex Data

[1] https://defuse.ca/online-x86-assembler.htm#disassembly

一. 微处理器的结构(8086/8088)

1. 8086/8088 CPU的结构




```
MOV     BL, 06H
MOV     CL, 05H
ADD     BL, CL
```

GCC Assembler [1]

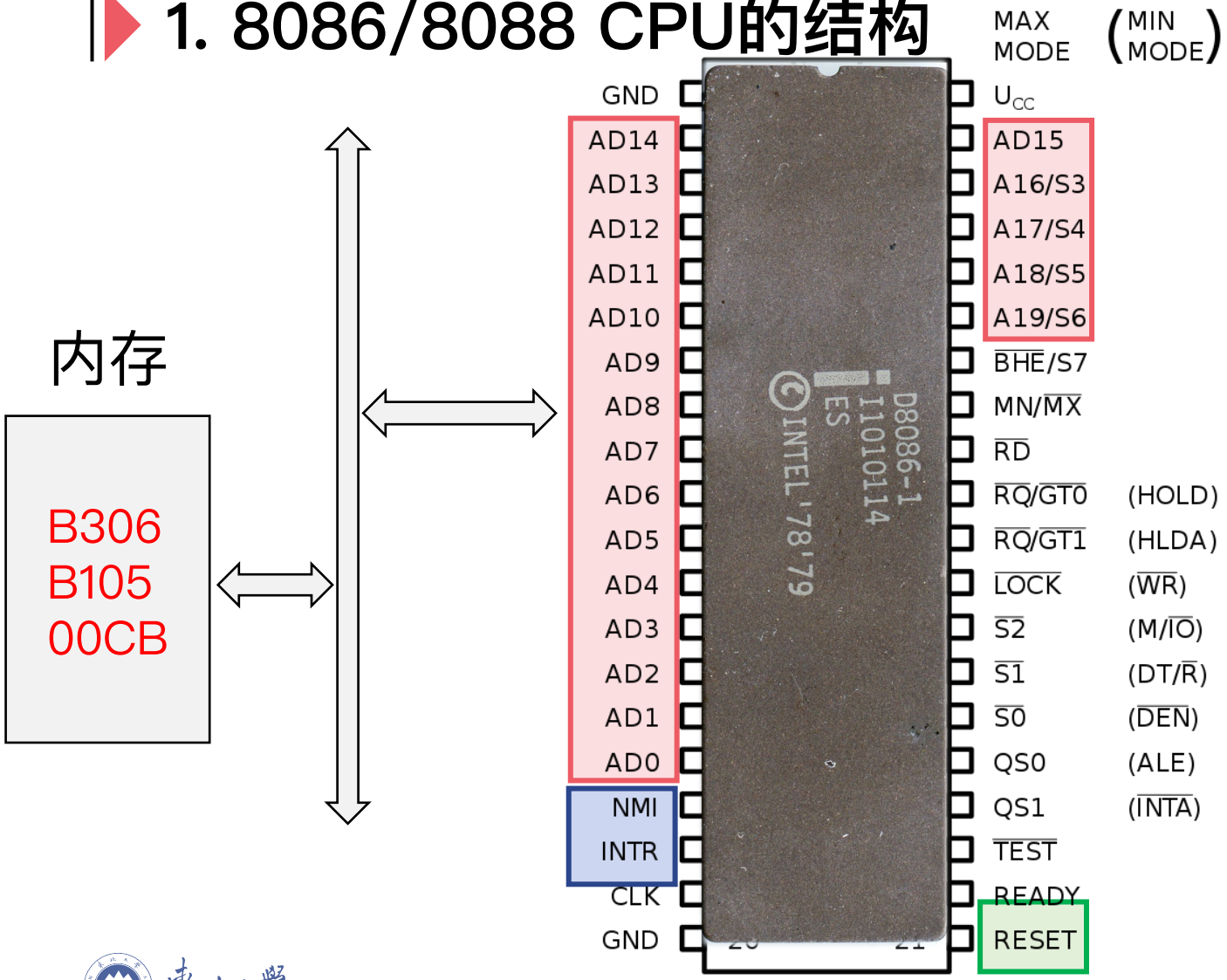
B306
B105
00CB

机器码
Raw Hex Data

[1] https://defuse.ca/online-x86-assembler.htm#disassembly

一. 微处理器的结构(8086/8088)

1. 8086/8088 CPU的结构




```
MOV     BL, 06H
MOV     CL, 05H
ADD     BL, CL
```

GCC Assembler [1]

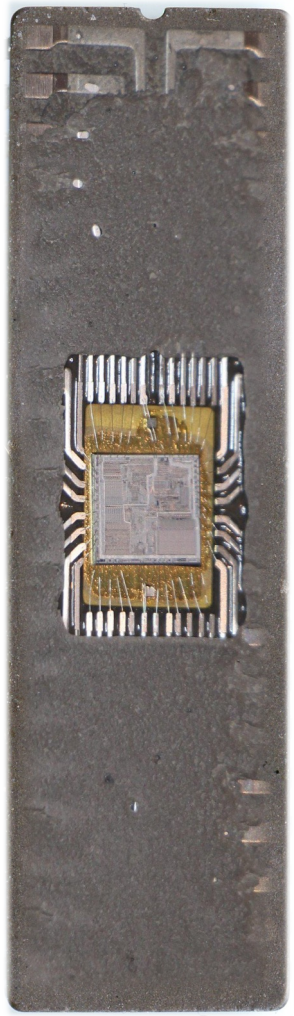
B306
00CB Raw Hex Data

具体是怎么工作的？

[1] https://defuse.ca/online-x86-assembler.htm#disassembly

一. 微处理器的结构(8086/8088)

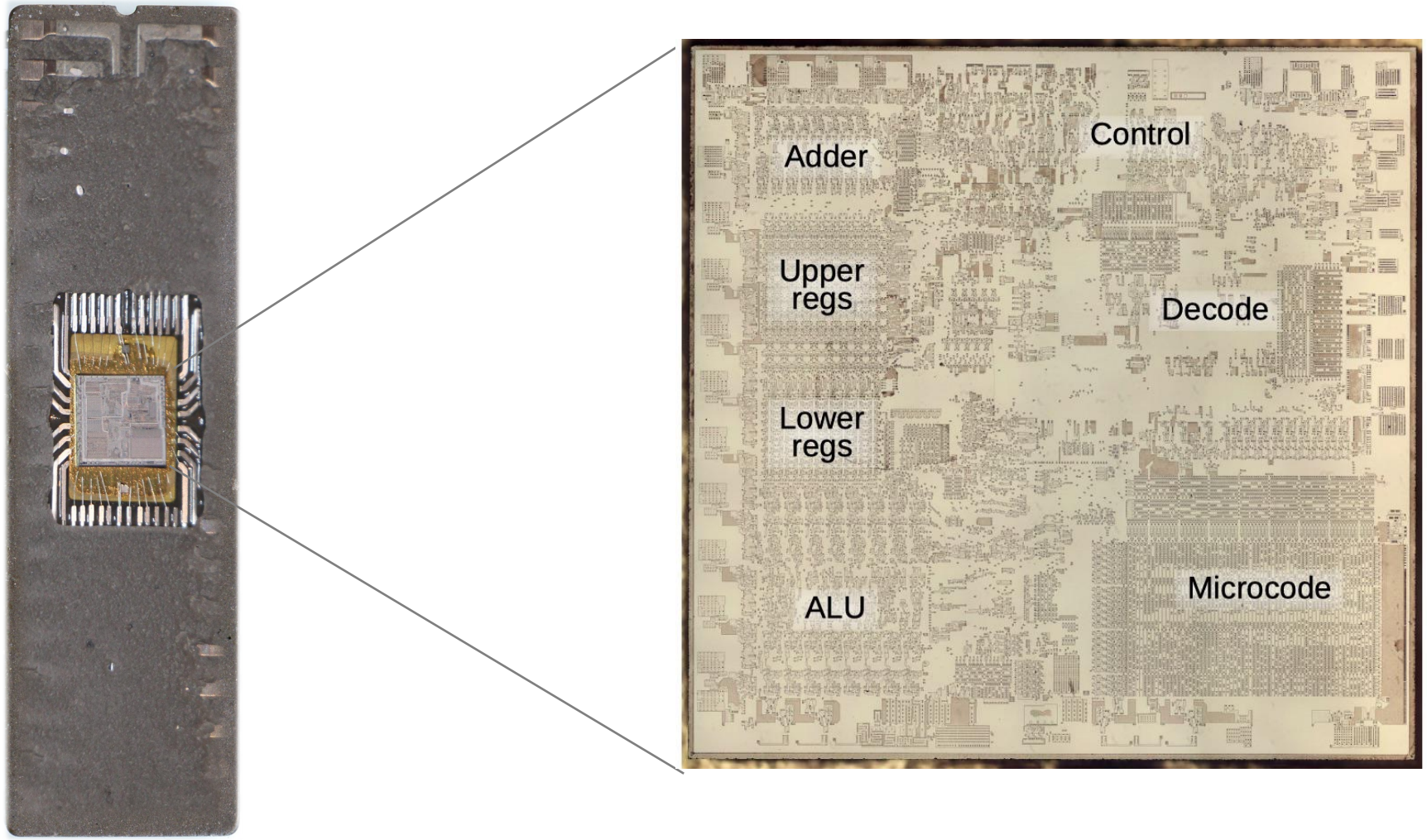
1. 8086/8088 CPU的结构



[1] https://www.engineering-sample.com/gallery/Intel/007%208086/D8086-1%20ES/
[2] http://www.righto.com/2020/06/a-look-at-die-of-8086-processor.html

一. 微处理器的结构(8086/8088)

1. 8086/8088 CPU的结构



[1] https://www.engineering-sample.com/gallery/Intel/007%208086/D8086-1%20ES/
[2] http://www.righto.com/2020/06/a-look-at-die-of-8086-processor.html

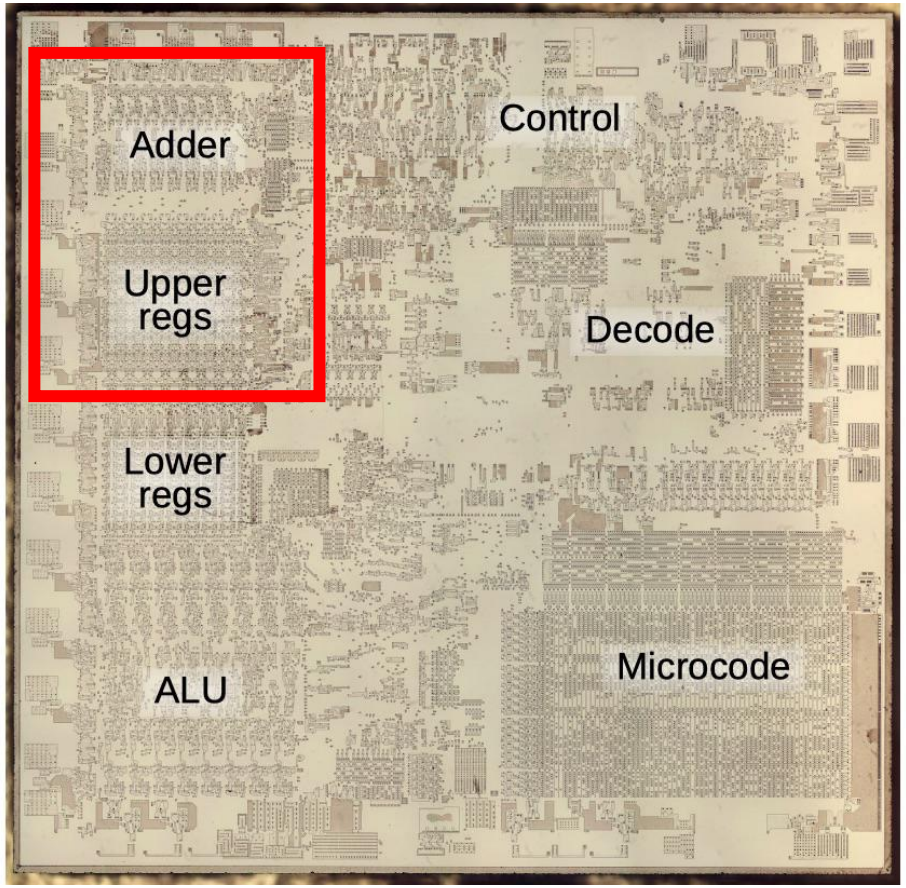
一. 微处理器的结构(8086/8088)

1. 8086/8088 CPU的结构

A look at the die of the 8086 processor [2]

The **left side** of the chip contains the 16-bit data path: the chip's registers and arithmetic circuitry. The adder and upper registers form the Bus Interface Unit that communicates with external memory, while the lower registers and the ALU form the Execution Unit that processes data. The right side of the chip has control circuitry and instruction decoding, along with the microcode ROM that controls each instruction.

Bus Interface Unit (BIU)

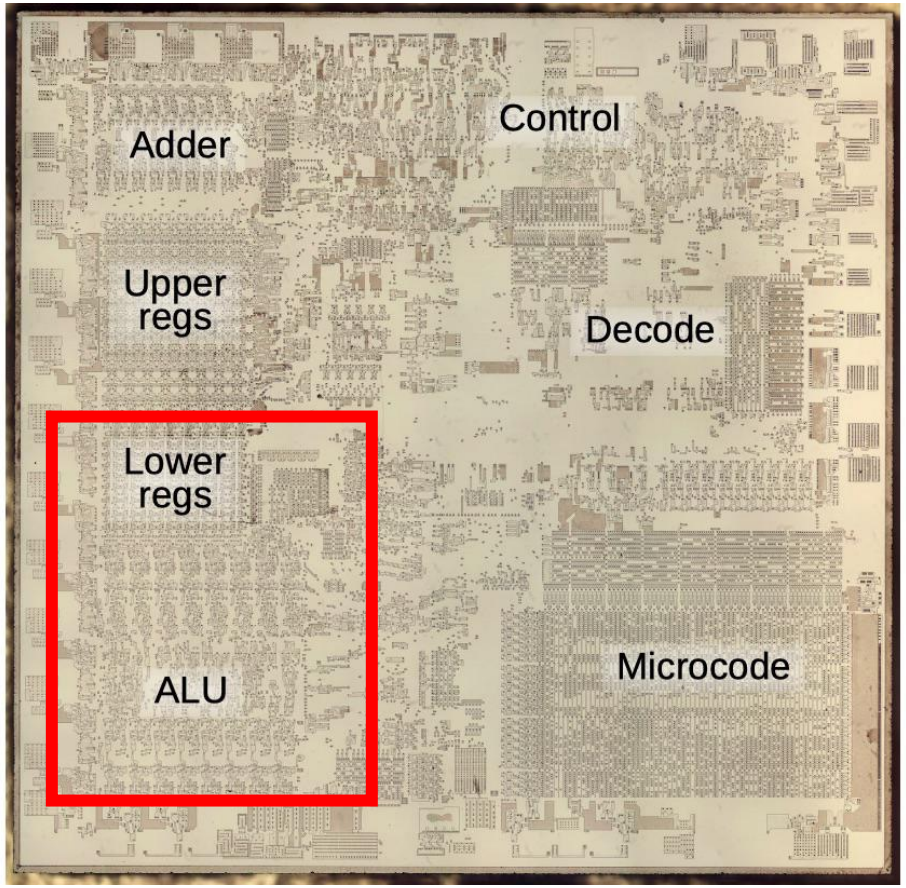


[1] https://www.engineering-sample.com/gallery/Intel/007%208086/D8086-1%20ES/
[2] https://www.righto.com/2020/06/a-look-at-die-of-8086-processor.html

1. 8086/8088 CPU的结构

A look at the die of the 8086 processor [2]

The **left side** of the chip contains the 16-bit data path: the chip's registers and arithmetic circuitry. The adder and upper registers form the Bus Interface Unit that communicates with external memory, while the lower registers and the ALU form the Execution Unit that processes data. The right side of the chip has control circuitry and instruction decoding, along with the microcode ROM that controls each instruction.



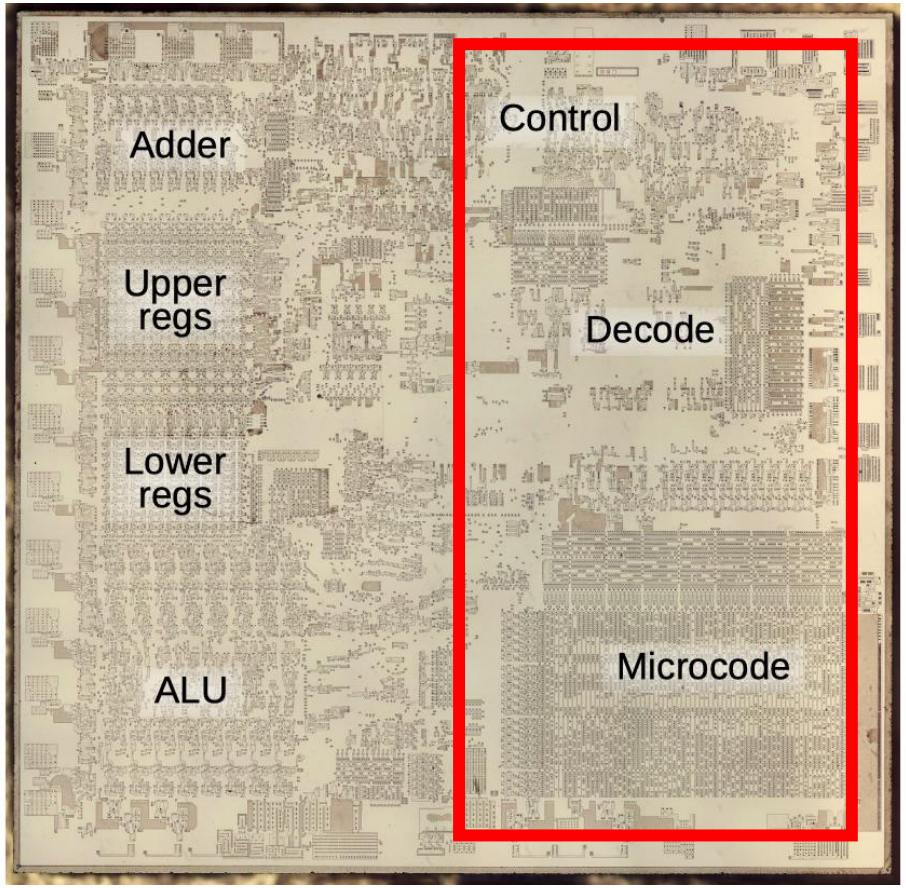
Execution Unit (EU)

[1] https://www.engineering-sample.com/gallery/Intel/007%208086/D8086-1%20ES/
[2] https://www.righto.com/2020/06/a-look-at-die-of-8086-processor.html

1. 8086/8088 CPU的结构

A look at the die of the 8086 processor [2]

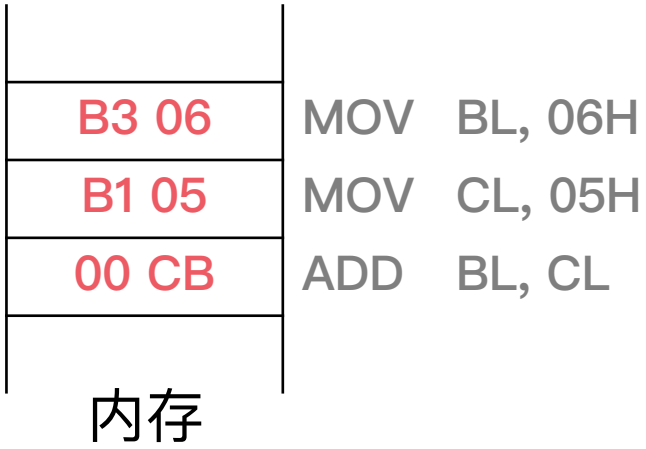
The **left side** of the chip contains the 16-bit data path: the chip's registers and arithmetic circuitry. The adder and upper registers form the Bus Interface Unit that communicates with external memory, while the lower registers and the ALU form the Execution Unit that processes data. The right side of the chip has control circuitry and instruction decoding, along with the microcode ROM that controls each instruction.



[1] https://www.engineering-sample.com/gallery/Intel/007%208086/D8086-1%20ES/
[2] https://www.righto.com/2020/06/a-look-at-die-of-8086-processor.html

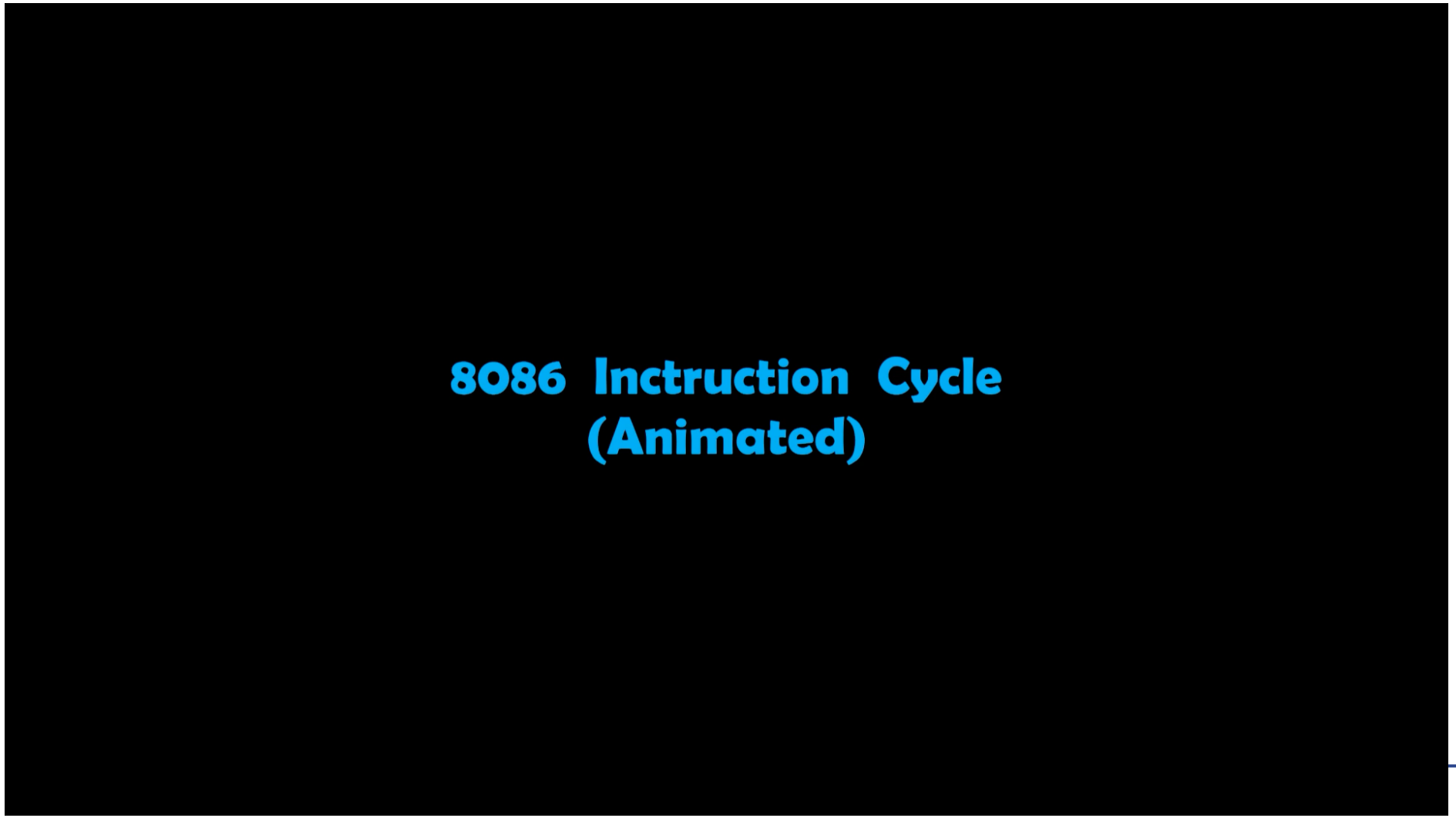
一. 微处理器的结构(8086/8088)

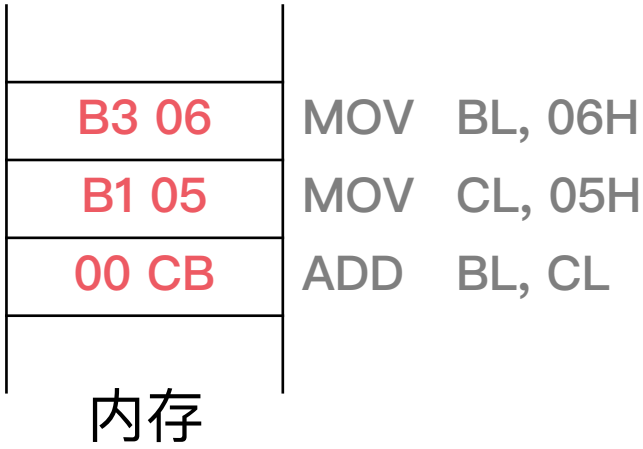
1. 8086/8088 CPU的结构



一. 微处理器的结构(8086/8088)

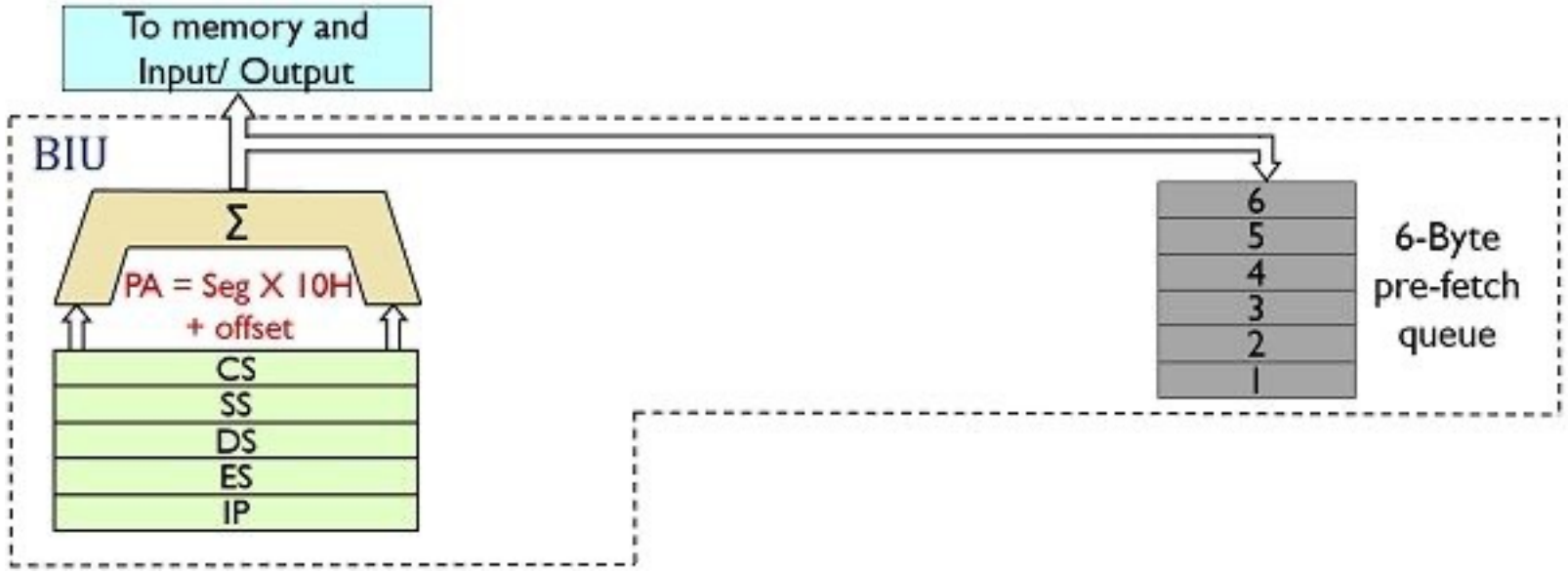
1. 8086/8088 CPU的结构





一. 微处理器的结构(8086/8088)

1. 8086/8088 CPU的结构

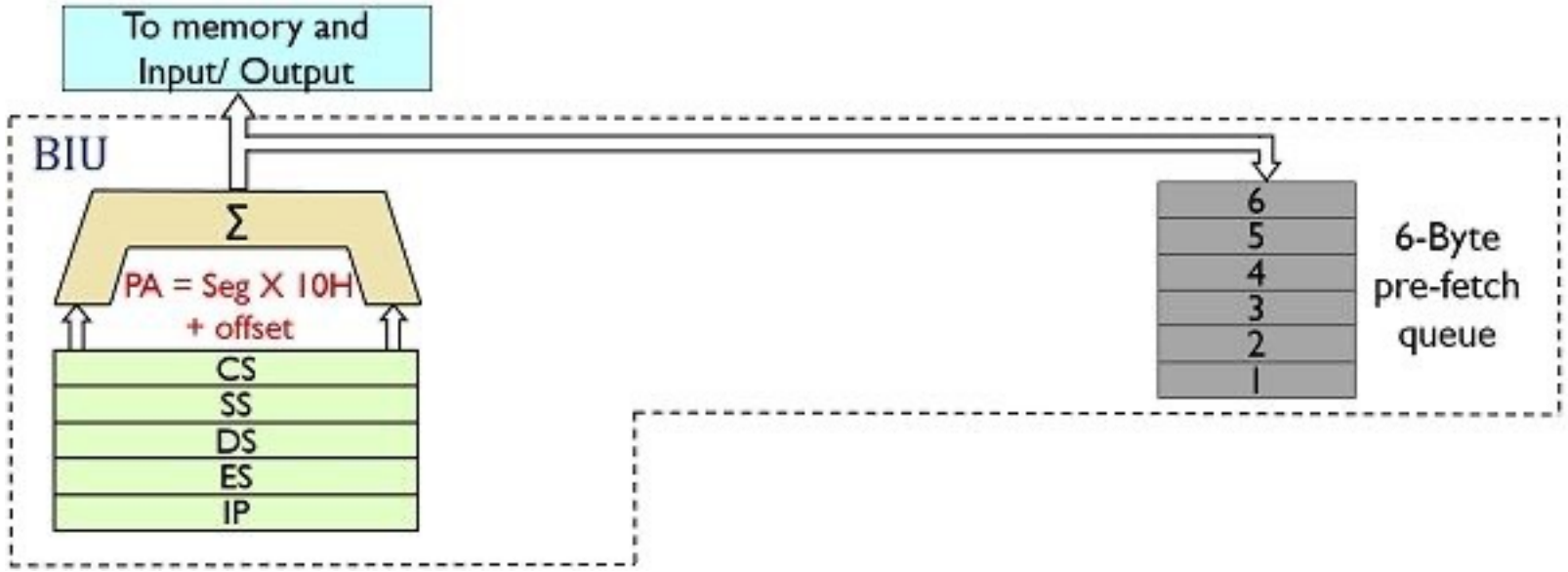


BIU:

Block Diagram of 8086 Microprocessor[1] [1] https://electronicsdesk.com/8086-microprocessor.html

一. 微处理器的结构(8086/8088)

1. 8086/8088 CPU的结构



BIU:

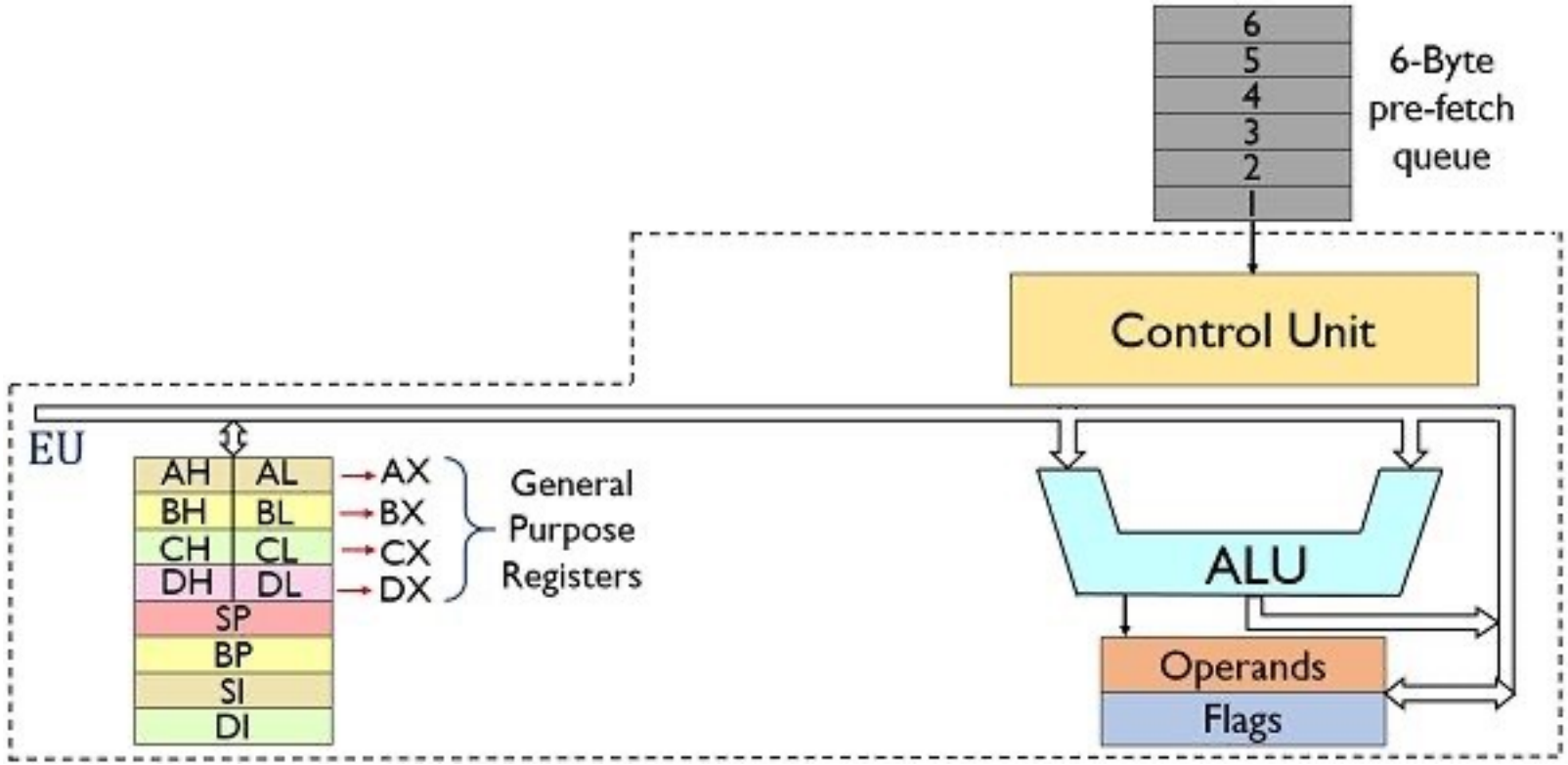
1. Generate 20-bit physical address for memory access
2. It fetches instructions from the memory;
3. It transfer data to and from the memory and I/O;
4. Maintains the 6-byte prefetch instruction

Block Diagram of 8086 Microprocessor[1] [1] https://electronicsdesk.com/8086-microprocessor.html

一. 微处理器的结构(8086/8088)

1. 8086/8088 CPU的结构

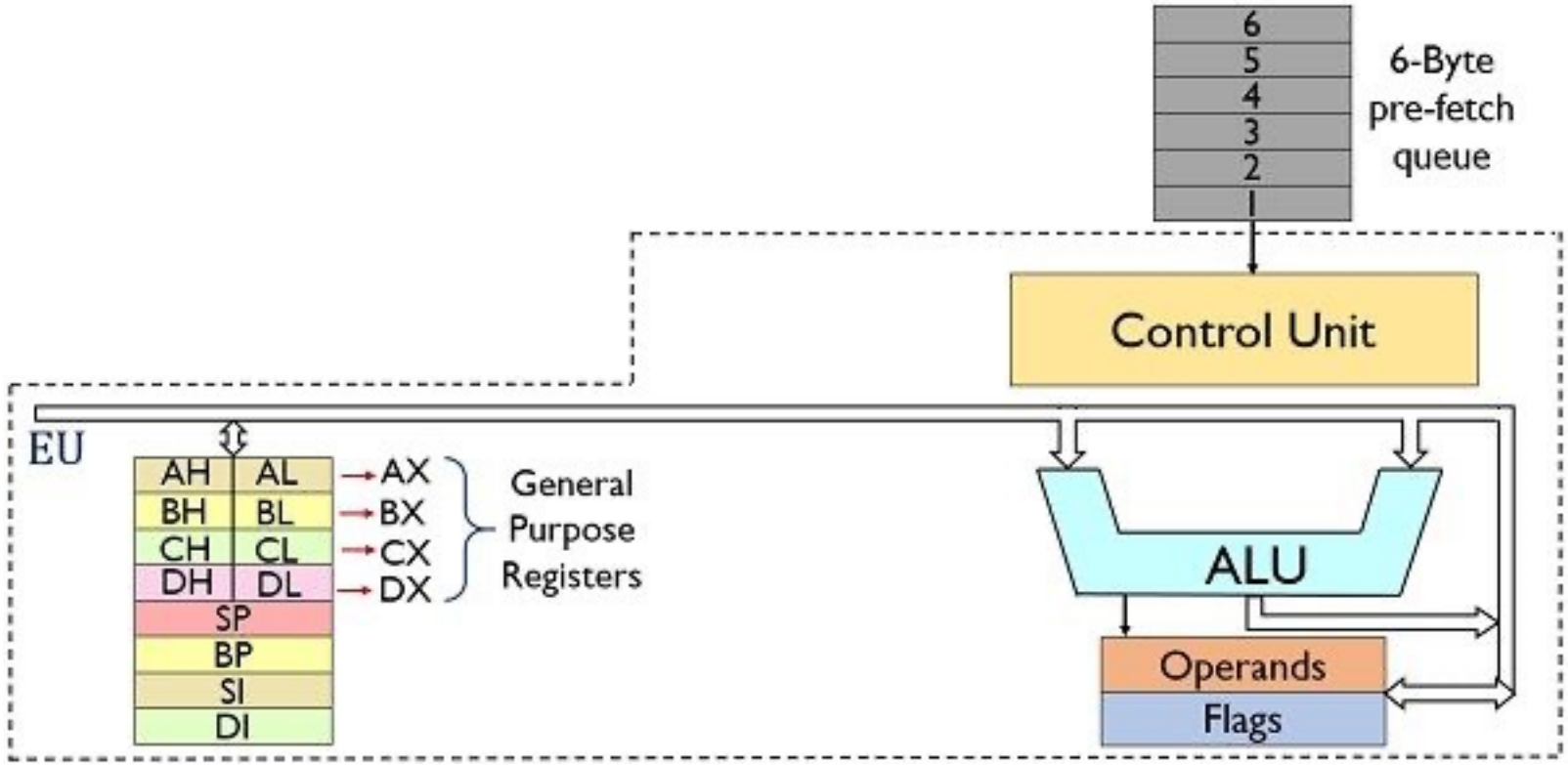
EU:



Block Diagram of 8086 Microprocessor[1] [1] https://electronicsdesk.com/8086-microprocessor.html

一. 微处理器的结构(8086/8088)

1. 8086/8088 CPU的结构



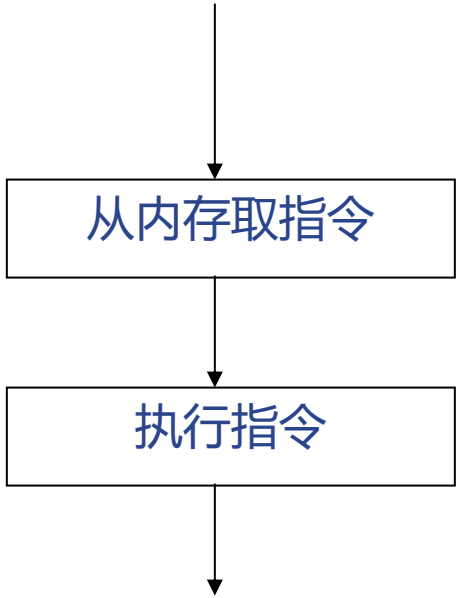
Block Diagram of 8086 Microprocessor[1] [1] https://electronicsdesk.com/8086-microprocessor.html

EU:

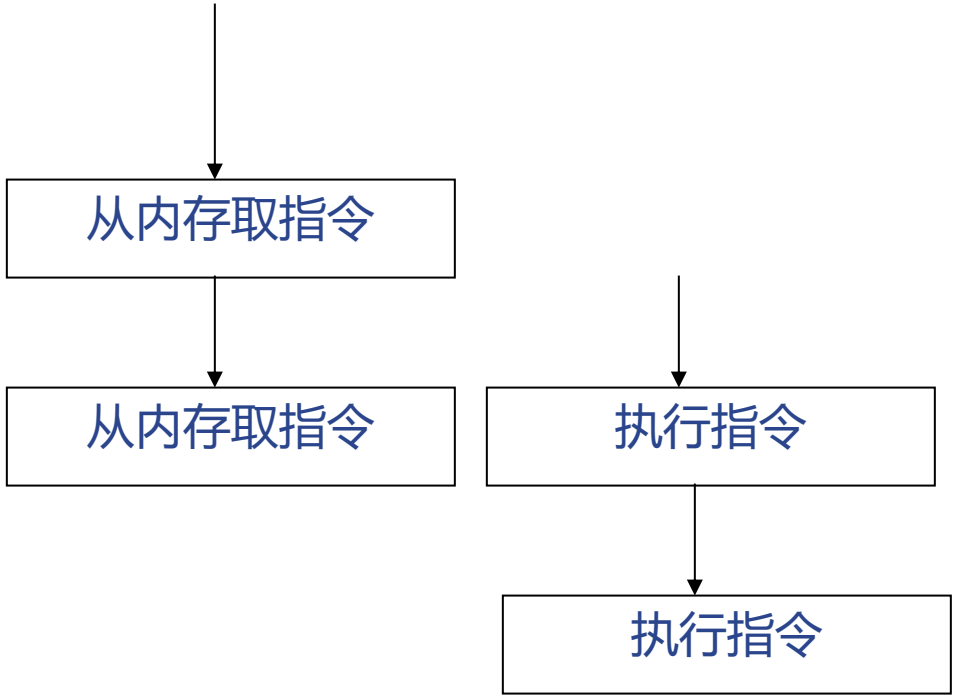
1. EU is responsible for the execution of instructions
2. It tells BIU from where to fetch instruction and data

一. 微处理器的结构(8086/8088)

流水线（pipeline）



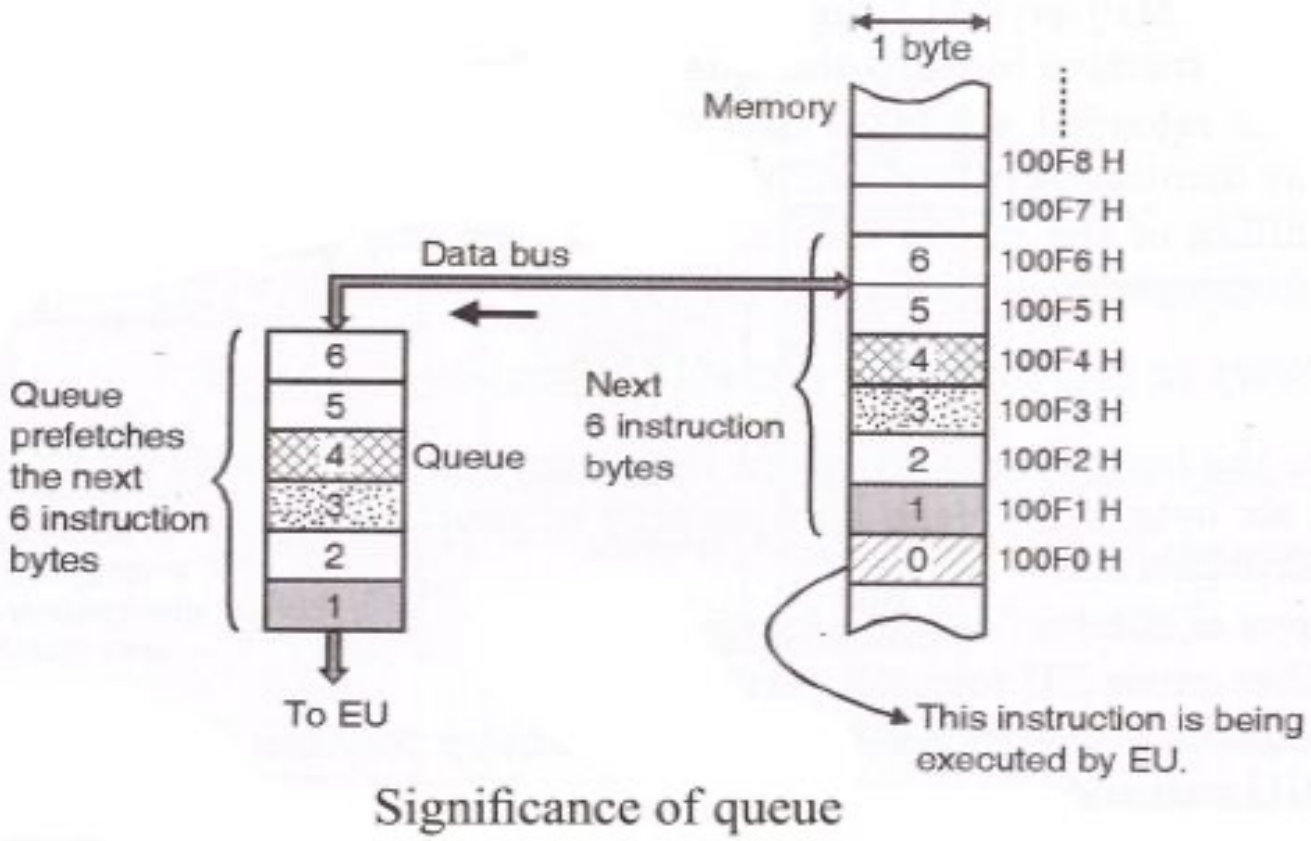
传统CPU执行指令的过程



现代CPU执行指令的过程

流水线（pipeline）[知识扩展]

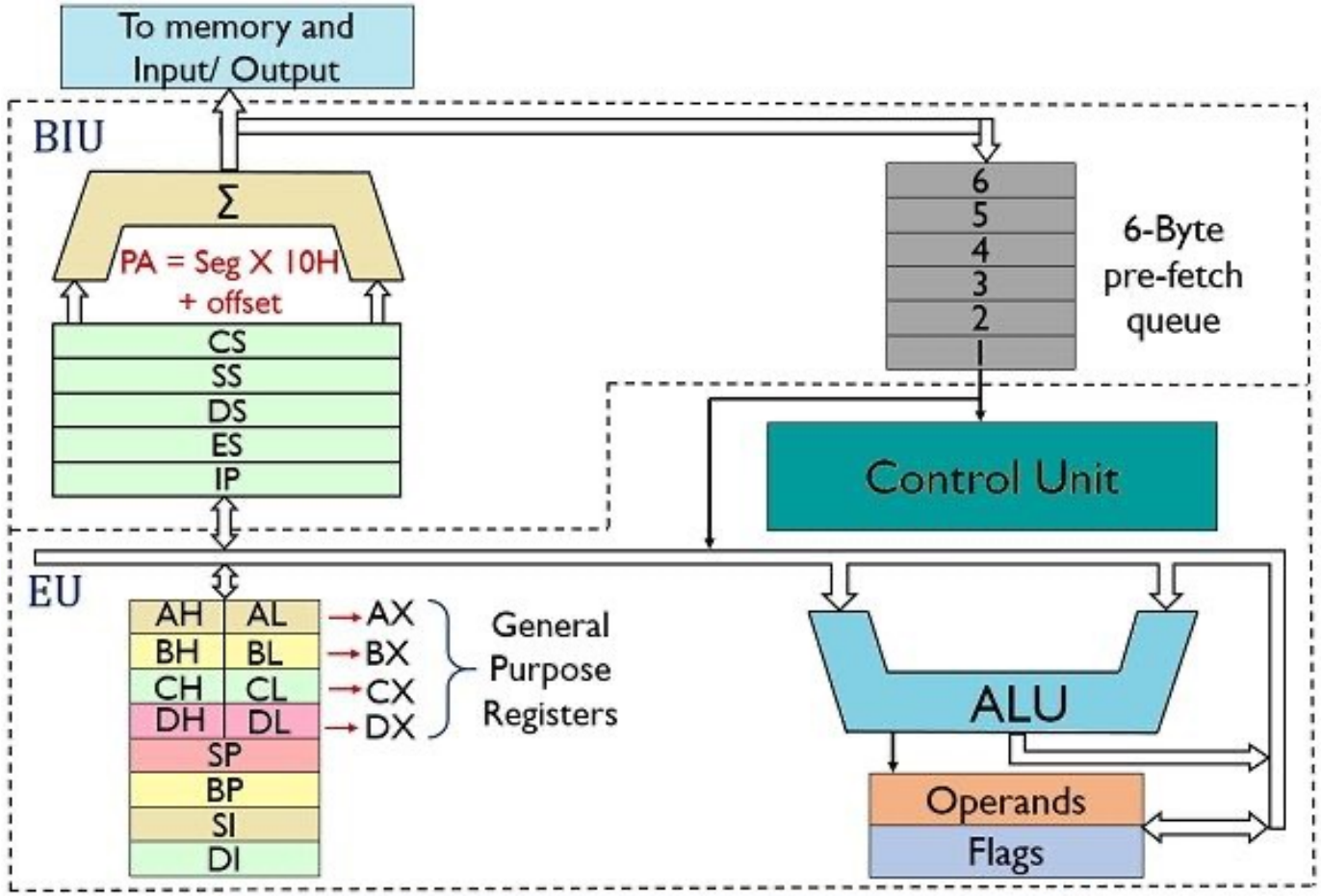
- The process of fetching the next instruction when the present instruction is being executed is called as pipelining
- Pipelining has become possible due to the use of queue.
- BIU (Bus Interfacing Unit) fills in the queue until the entire queue is full.
- BIU restarts filling in the queue when at least two locations of queue are vacant.
- The 8086 BIU will not initiate a fetch unless and until there are two empty bytes in its queue
- 8086 BIU normally obtains two instruction bytes per fetch.



Significance of queue

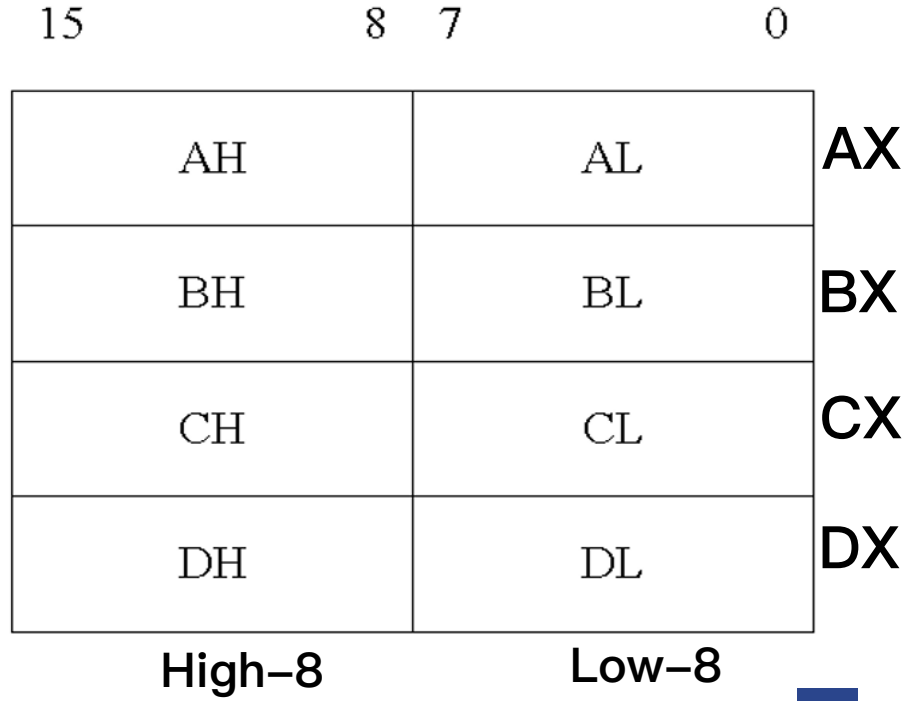
2. 8086/8088CPU的寄存器（Register）

- 通用寄存器
- 指针和变址寄存器
- 段寄存器
- 标志寄存器



2. 8086/8088CPU的寄存器（四种类型）

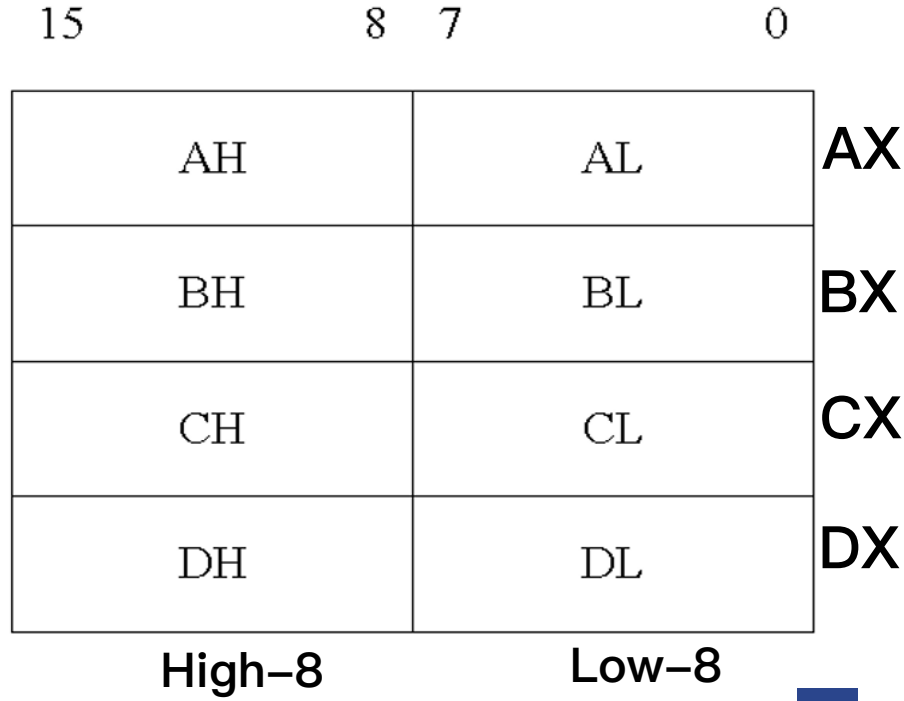
- **通用寄存器**：主要用于算数运算和数据传输
  - ➢每个通用寄存器可以分为两个8位寄存器，单独使用
  - ➢每个通用寄存器，一般用途时可以互换



2. 8086/8088CPU的寄存器（四种类型）

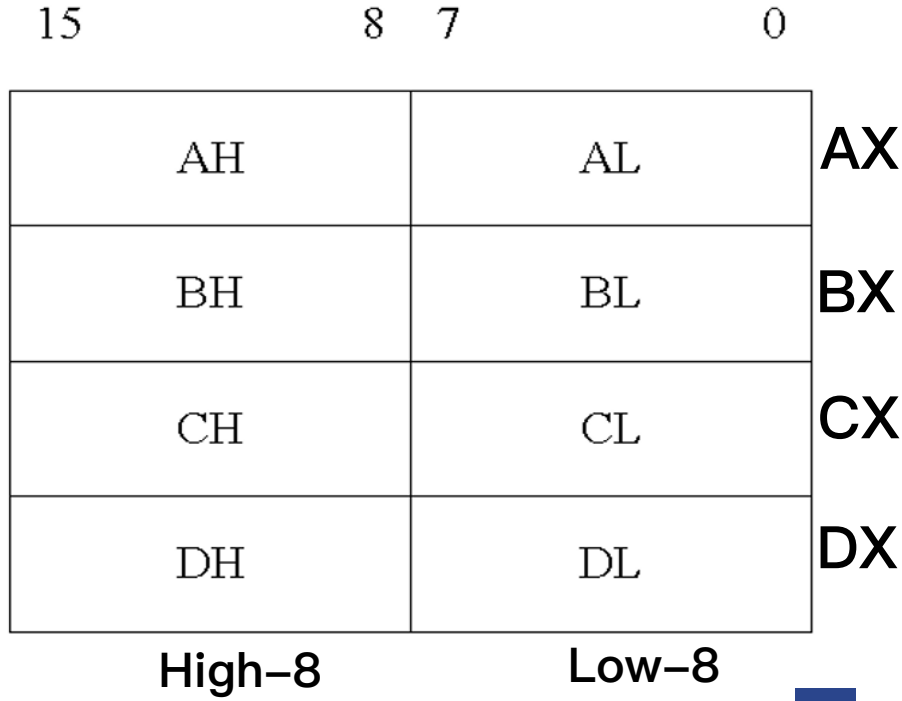
- **通用寄存器**：主要用于算数运算和数据传输
  - ➢每个通用寄存器可以分为两个8位寄存器，单独使用
  - ➢每个通用寄存器，一般用途时可以互换

```
MOV AX,1
MOV BX,2
ADD AX,BX ;AX<--AX+BX ,结果AX=3
```



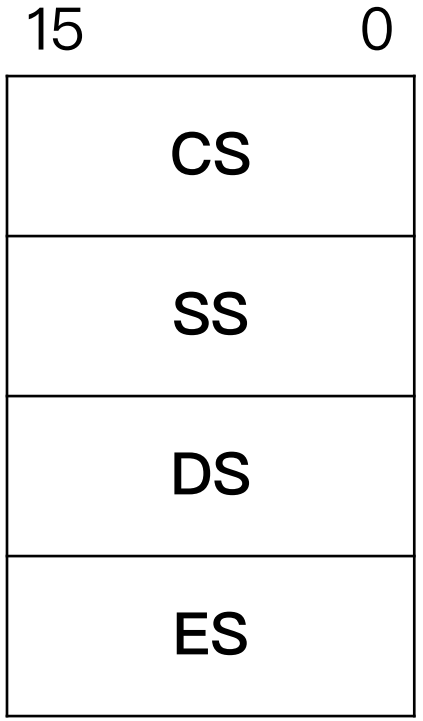
2. 8086/8088CPU的寄存器（四种类型）

- **通用寄存器**：主要用于算数运算和数据传输
  - ➢每个通用寄存器可以分为两个8位寄存器，单独使用
  - ➢每个通用寄存器，一般用途时可以互换
  - ➢少量指令会把某些寄存器作为专用
    - AX(Accumulator)：主累加器（乘除法指令）
    - BX (Base)：基址寄存器（基址寻址）
    - CX (Count)：计数器寄存器（循环指令）
    - DX (Data)：数据寄存器（I/O地址存储）



2. 8086/8088CPU的寄存器（Register）

- **段寄存器（Segment Register）**
  - 8086CPU具有寻址1MB的能力，但寄存器是16位的，不能直接寻址1MB空间。
  - 将1MB空间划分成若干逻辑段，每段≤64KB
  - 解释：
    - CS：存储代码段基地址
    - SS：存储堆栈段基地址
    - DS：存储数据段基地址
    - ES：存储附加段基地址



代码段寄存器
(Code Segment Register)

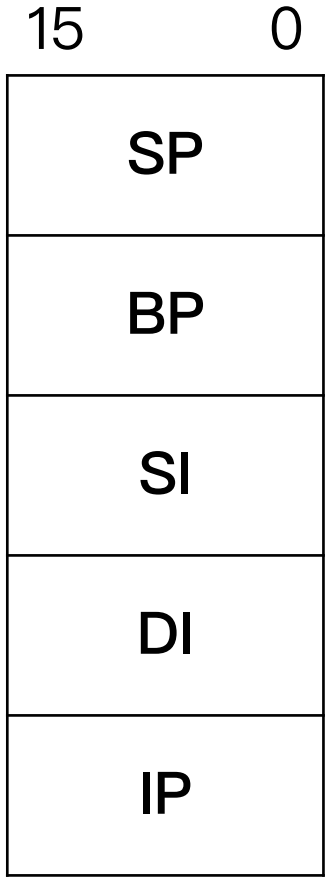
堆栈段寄存器
(Stack Segment Register)

数据段寄存器
(Data Segment Register)

附加段寄存器
(Extra Segment Register)

2. 8086/8088CPU的寄存器（Register）

- 指针和变址寄存器（特殊功能寄存器）



堆栈指针
(Stack Pointer)

基址指针
(Base Pointer)

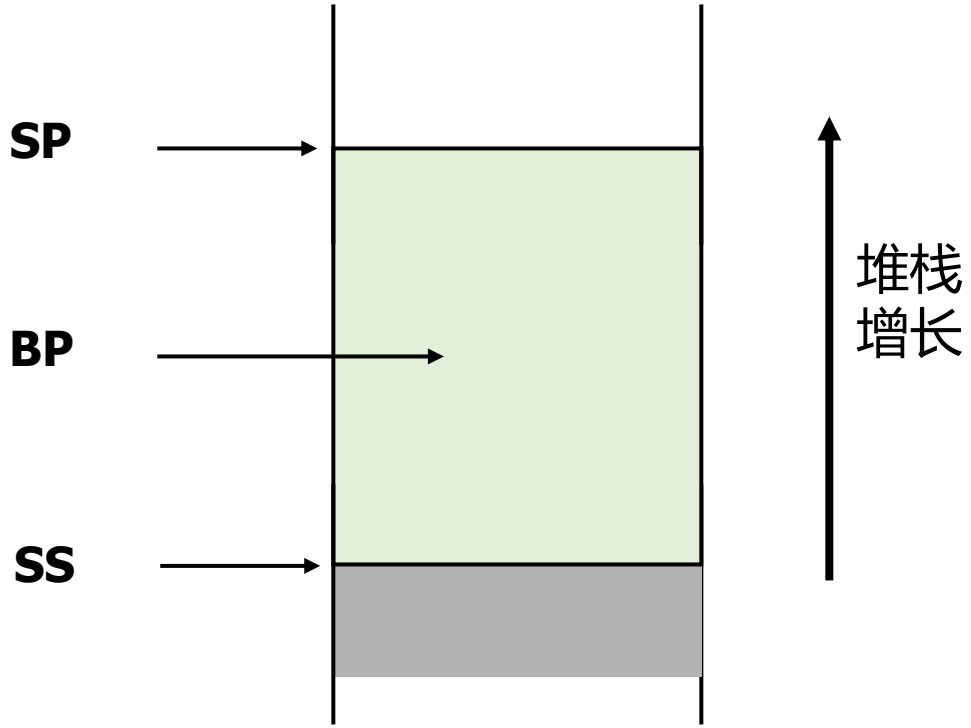
源变址寄存器
(Source Index Register)

目的变址寄存器
(Destination Index Register)

指令指针
(Instruction Pointer)

2. 8086/8088CPU的寄存器（Register）

- **指针和变址寄存器（特殊功能寄存器）**
  - SP和BP都用来指示当前堆栈段中数据所在的地址，但有所区别
    - SP：指示栈顶
    - BP：作为堆栈段的内存指针，指示任一位置



2. 8086/8088CPU的寄存器（Register）

- **指针和变址寄存器（特殊功能寄存器）**
  - SI和DI存放当前数据段的偏移地址
    - SI：存放源操作数的偏移地址
    - DI：存放目标操作数的偏移地址

2. 8086/8088CPU的寄存器（Register）

- **指针和变址寄存器（特殊功能寄存器）**
  - IP–指令指针寄存器
    - 下一条要执行指令的偏移地址
    - 不能使用普通传值改变IP寄存器的值
    - 可以通过子程序调用/返回、中断调用/返回等指令改变

2. 8086/8088CPU的寄存器

- **标志寄存器* （F)**
  - 标志寄存器长度为16位，其中9位有定义

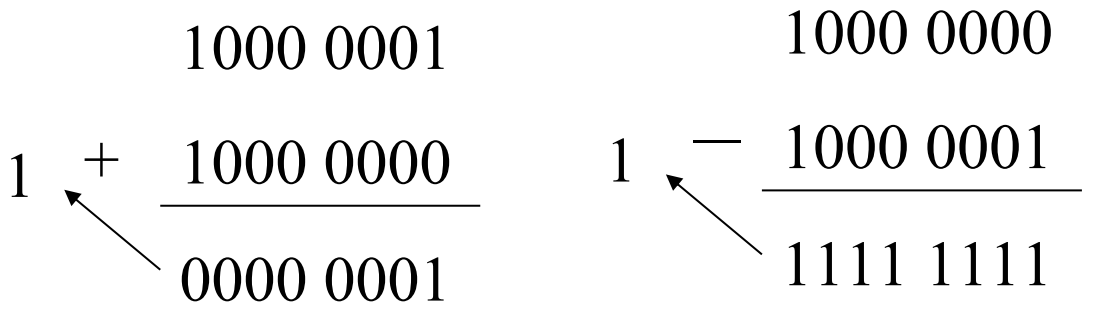
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	OF	DF	IF	TF	SF	ZF	X	AF	X	PF	X	CF

2. 8086/8088CPU的寄存器

- **标志寄存器*（F）**
  - 标志寄存器长度为16位，其中9位有定义

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	OF	DF	IF	TF	SF	ZF	X	AF	X	PF	X	CF

CF, Carry Flag, 进位标志。
两数相加,最高位向前的进位;
或两数相减最高位向前的借位



*：标志位在汇编语言程序设计中非常重要，必须掌握好

2. 8086/8088CPU的寄存器

- **标志寄存器*（F）**
  - 标志寄存器长度为16位，其中9位有定义

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	OF	DF	IF	TF	SF	ZF	X	AF	X	PF	X	CF

AF, Auxiliary Carry Flag, 辅助进位标志, 两数相加,第3位向前的进位; 或两数相减第3位向前的借位

$$\begin{array}{r} 0000\ 1001 \\ +\ 0001\ 1000 \\ \hline 0001\ 0001 \end{array} \qquad \begin{array}{r} 0000\ 0000 \\ -\ 0000\ 0001 \\ \hline 1111\ 1111 \end{array}$$

*：标志位在汇编语言程序设计中非常重要，必须掌握好

2. 8086/8088CPU的寄存器

- **标志寄存器*（F）**
  - 标志寄存器长度为16位，其中9位有定义

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	OF	DF	IF	TF	SF	ZF	X	AF	X	PF	X	CF

PF，Parity Flag，奇偶标志，两数操作(算术或逻辑),结果的低8位中含有1的位数是偶还是奇

1000 0101 1010 1100

AND 1001 1101 1001 0111

1000 0101 1000 0100

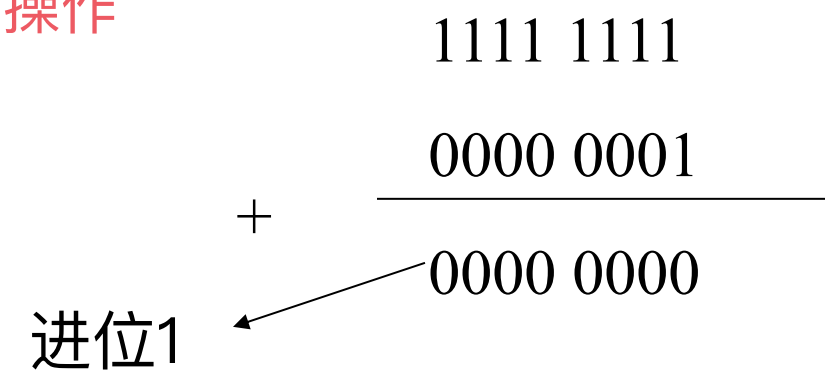
*：标志位在汇编语言程序设计中非常重要，必须掌握好

2. 8086/8088CPU的寄存器

- **标志寄存器*（F）**
  - 标志寄存器长度为16位，其中9位有定义

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	OF	DF	IF	TF	SF	ZF	X	AF	X	PF	X	CF

ZF, Zero Flag, 零标志, 操作结果的为0则ZF为1



操作结果为0, ZF=1

*：标志位在汇编语言程序设计中非常重要，必须掌握好

2. 8086/8088CPU的寄存器

- **标志寄存器*（F）**
  - 标志寄存器长度为16位，其中9位有定义

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	OF	DF	IF	TF	SF	ZF	X	AF	X	PF	X	CF

SF, Sign Flag, 符号标志, 操作结果的符号位(即最高位的状态)

1000 0101 1010 1100

AND 1001 1101 1001 0111

1000 0101 1000 0100

结果: SF=1

*：标志位在汇编语言程序设计中非常重要，必须掌握好

2. 8086/8088CPU的寄存器

- **标志寄存器* （F)**
  - 标志寄存器长度为16位，其中9位有定义

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	OF	DF	IF	TF	SF	ZF	X	AF	X	PF	X	CF

OF, Overflow Flag, 溢出标志, 算术操作,结果超过目标所能容纳的范围

$$\begin{array}{rll} & 0111\ 1110 & (+126) \\ + & 0000\ 0011 & (+3) \\ \hline & 1000\ 0001 & (-127) \end{array}$$

*：标志位在汇编语言程序设计中非常重要，必须掌握好

2. 8086/8088CPU的寄存器

- **标志寄存器*（F）**
  - 标志寄存器长度为16位，其中9位有定义

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	OF	DF	IF	TF	SF	ZF	X	AF	X	PF	X	CF

*：标志位在汇编语言程序设计中非常重要，必须掌握好

2. 8086/8088CPU的寄存器

- **标志寄存器* （F)**
  - 例子：计算下列各表达式的值， 并且根据计算结果分析OF、SF、 ZF、AF、PF和CF各标志位的状态。
    1) 58H+63H
    2) 80H+90H
    3) FFH+FEH
    4) 38H+2BH
    5) 9CH–45H
    6) FEH–FFH

2. 8086/8088CPU的寄存器

- **标志寄存器* （F)**

➢ 例子：计算下列各表达式的值， 并且根据计算结果分析OF、SF、 ZF、AF、PF和CF各标志位的状态。

1) 58H+63H
2) 80H+90H
3) FFH+FEH
4) 38H+2BH
5) 9CH–45H
6) FEH–FFH

	OF	SF	ZF	AF	PF	CF	结果
58H+63H	1	1	0	0	1	0	BBH
80H+90H	1	0	0	0	0	1	10H
FFH+FEH	0	1	0	1	0	1	FDH
38H+2BH	0	0	0	1	1	0	63H
9CH–45H	1	0	0	0	0	0	57H
FEH–FFH	0	1	0	1	1	1	FFH

*：标志位在汇编语言程序设计中非常重要，必须掌握好

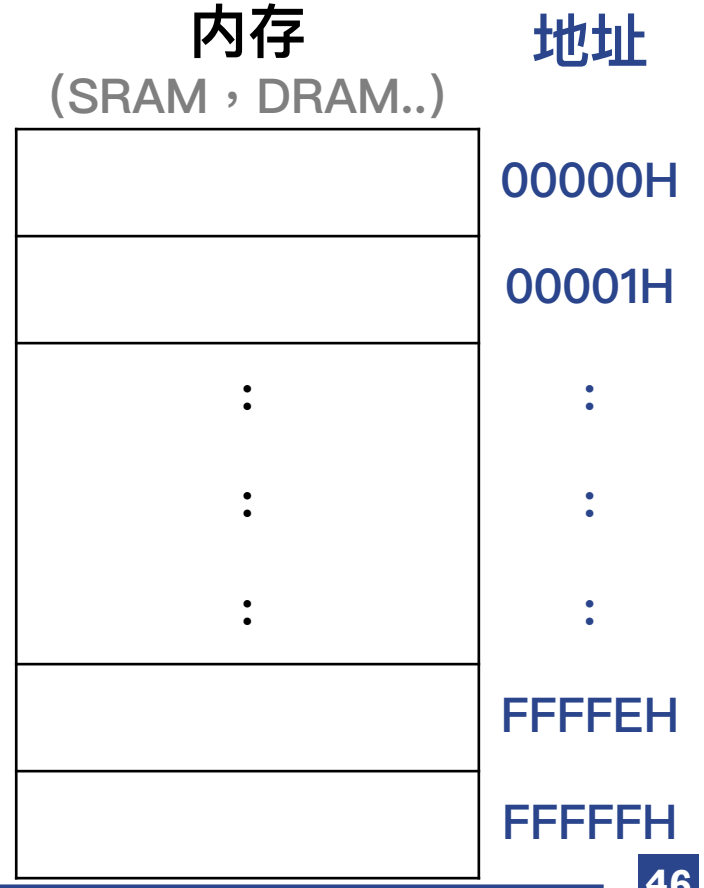
第三章 微型计算机的结构

二. 存储器(组织与结构)

1. 存储器的分段结构
2. 实际地址的产生

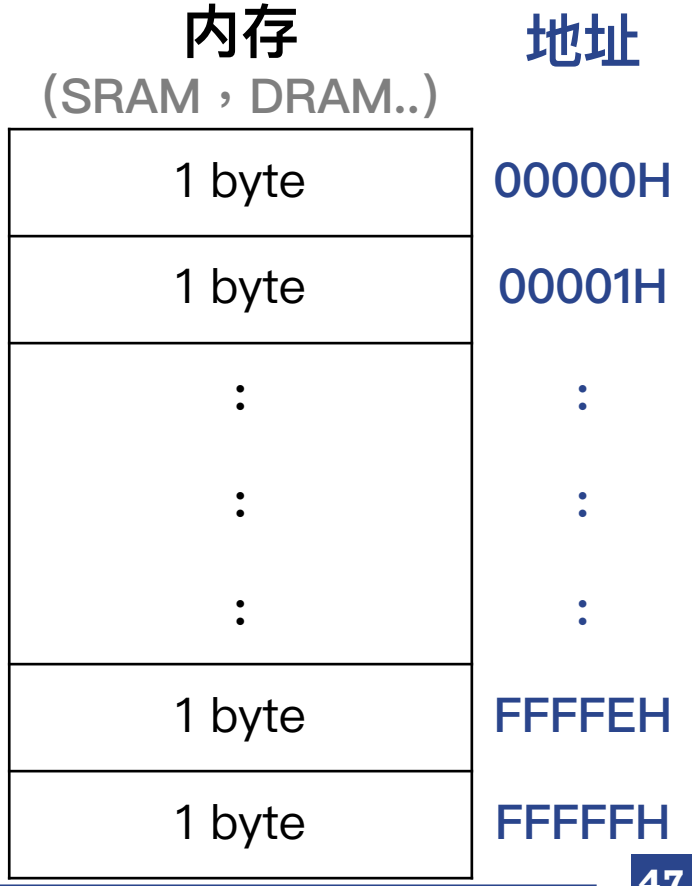
1. 存储器的分段结构

- 编址：8086CPU的地址总线20根 → 提供1M个地址



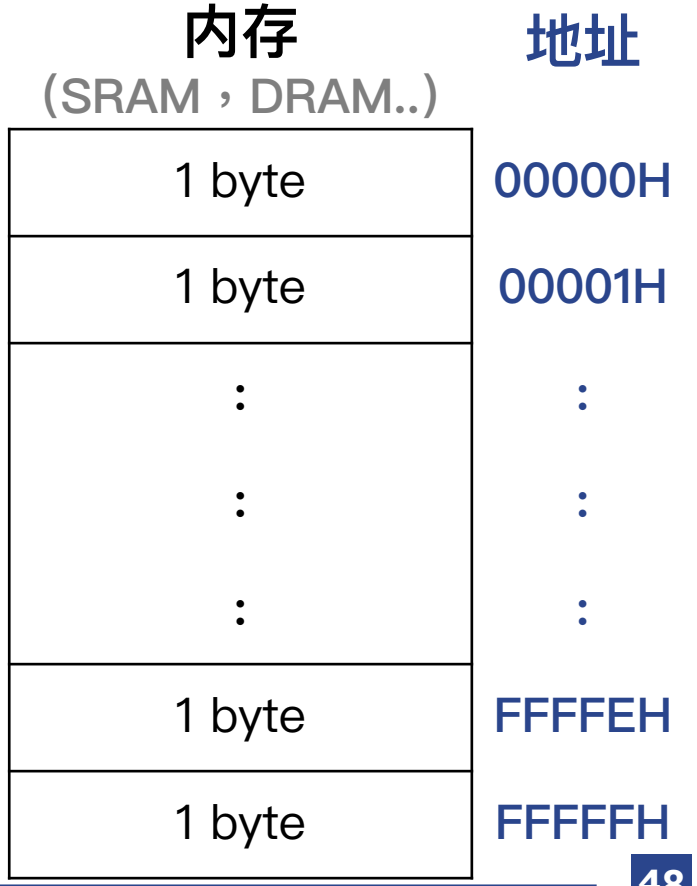
1. 存储器的分段结构

- 编址：8086CPU的地址总线20根 → 提供1M个地址
  - 8086能直接访问的最大内存大小1MB (Mega-byte)
    - 思考题：为什么是1MB？



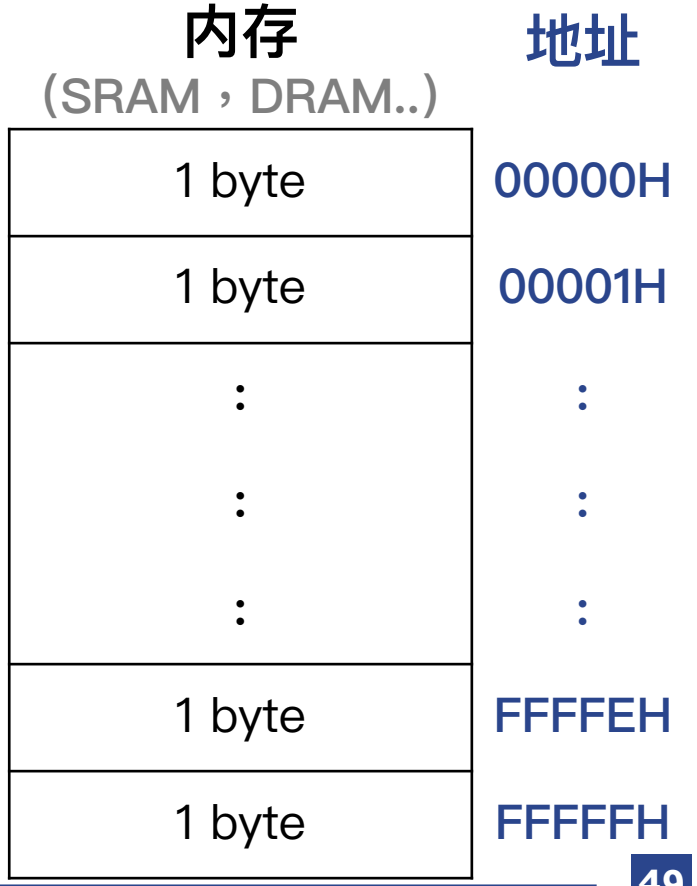
1. 存储器的分段结构

- 编址：8086CPU的地址总线20根 → 提供1M个地址
  - 8086能直接访问的最大内存大小1MB (Mega-byte)
  - 存储单位：
    - B (byte) = 8 (bits)
    - KB (Kilo-byte), 1KB = 1024B
    - MB (Mega-byte)
    - GB (Giga-byte)
    - TB (Tera-byte)



1. 存储器的分段结构

- 编址：8086CPU的地址总线20根 → 提供1M个地址
  - 8086能直接访问的最大内存大小1MB (Mega-byte)
  - 存储单位：
  - 能寻址1M个地址不意味有1MB内存

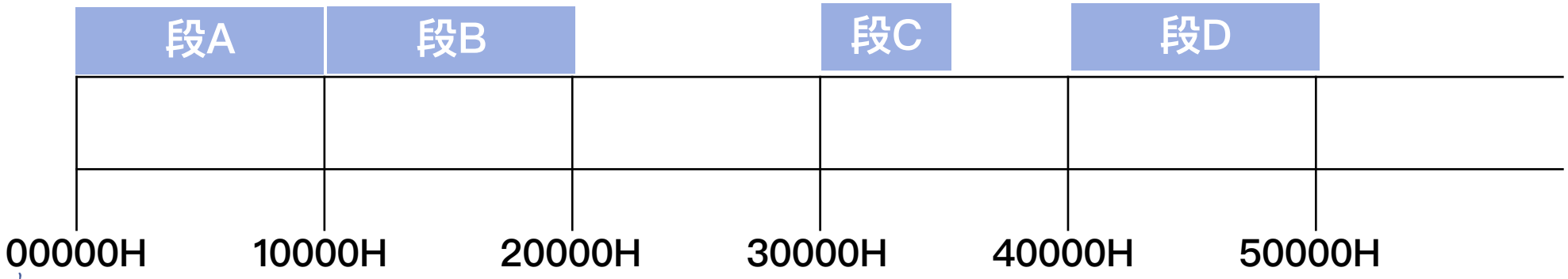


1. 存储器的分段结构

- 寻址：8086CPU寄存器16 bits，地址数据20 bits，无法直接存储
  - 16根地址线 → 2^{16} = 64KB
  - (1/16) * 1MB

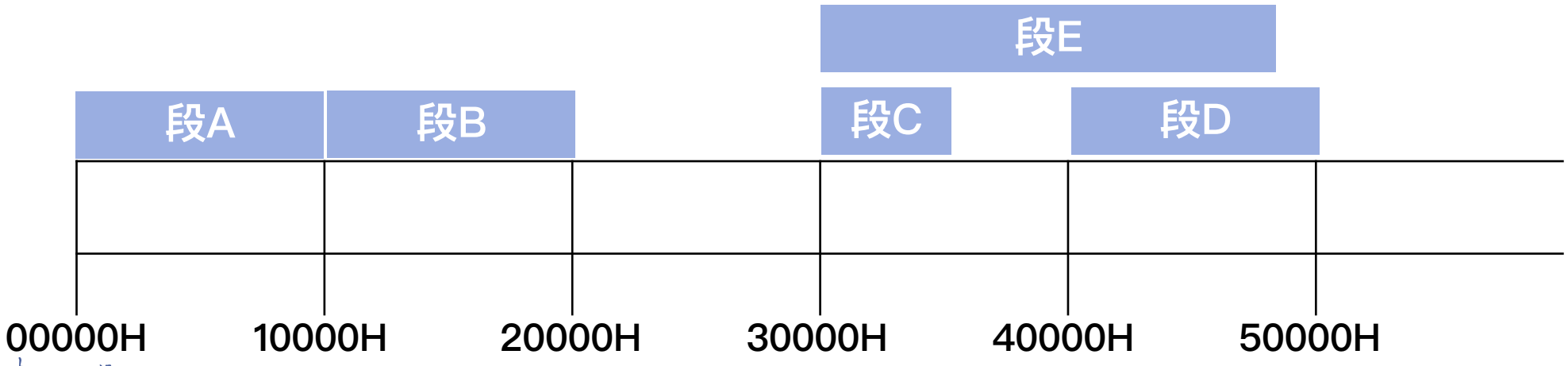
1. 存储器的分段结构

- 寻址：8086CPU寄存器16 bits，地址数据20 bits，无法直接存储
  - **存储器分段技术**：
    - 将1M的地址空间，逻辑上分成多个段
    - 每个段最大长度 ≤ 64KB



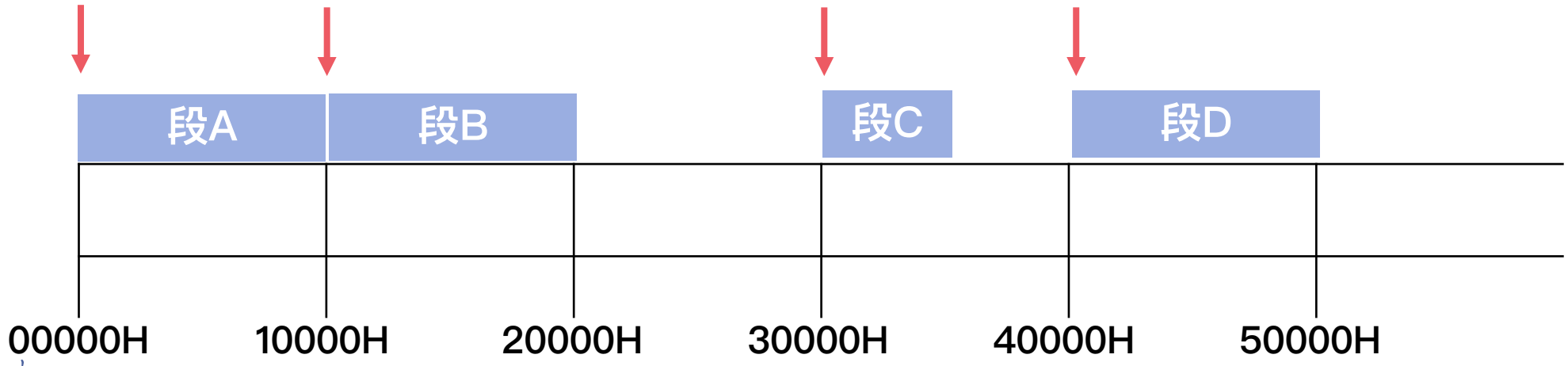
1. 存储器的分段结构

- 寻址：8086CPU寄存器16 bits，地址数据20 bits，无法直接存储
  - 存储器分段技术：
    - 段与段之间可以部分/全部重叠



1. 存储器的分段结构

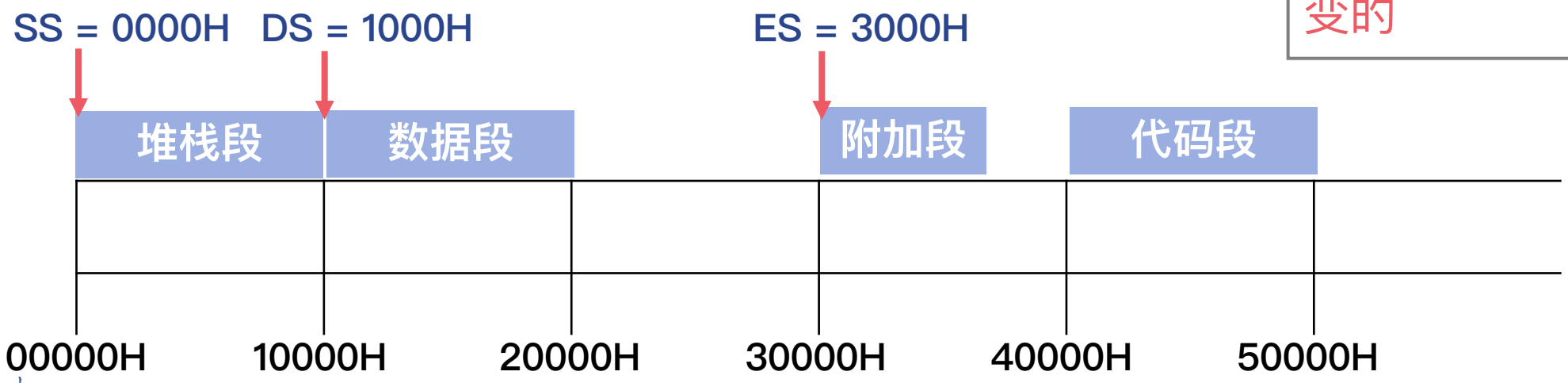
- 寻址：8086CPU寄存器16 bits，地址数据20 bits，无法直接存储
  - **存储器分段技术**：
    - 规定：段基址能够被16D整除



1. 存储器的分段结构

- 寻址：8086CPU寄存器16 bits，地址数据20 bits，无法直接存储
  - **存储器分段技术**：
    - 规定：段基址能够被16D整除
    - 段基址的存放：实际地址的高16位存放在CPU段寄存器中

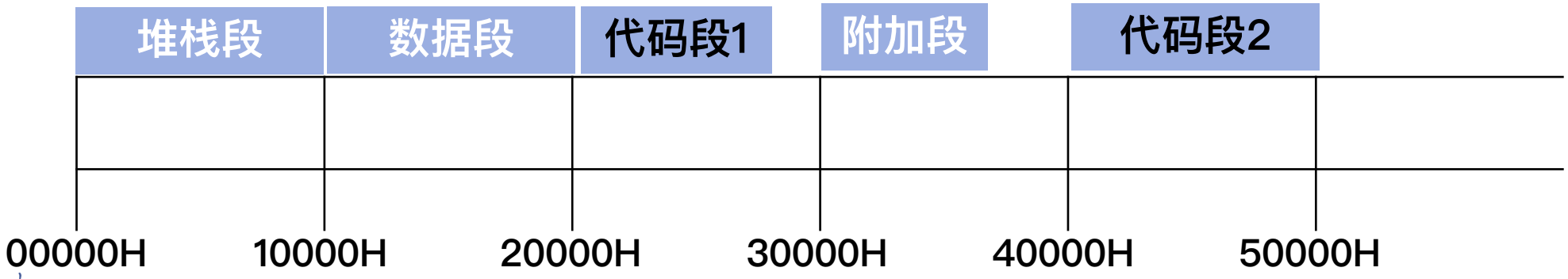
注意：SS、DS、ES的值是在程序执行过程中由传送指令（程序员）显式改变的



1. 存储器的分段结构

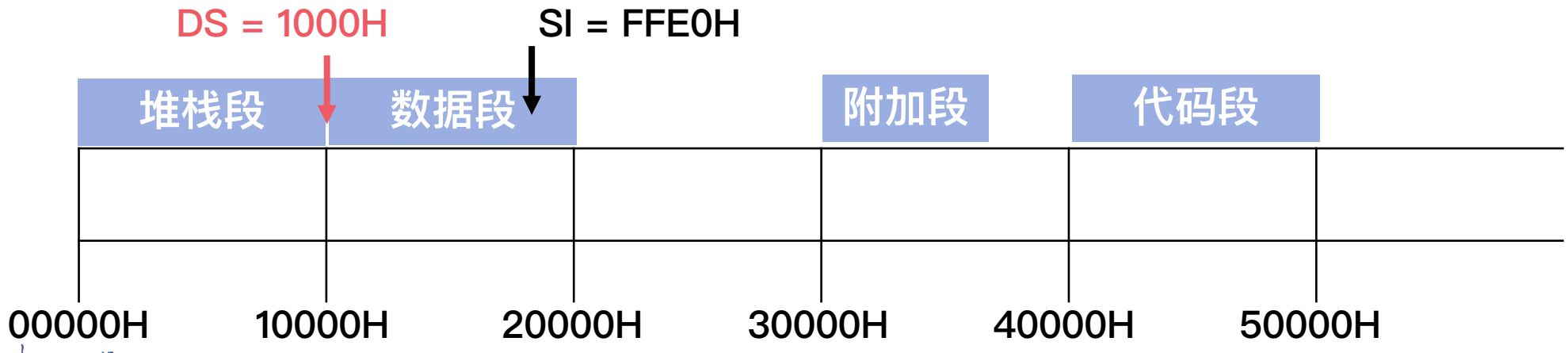
- 寻址：8086CPU寄存器16 bits，地址数据20 bits，无法直接存储
  - **存储器分段技术**：
    - 规定：段基址能够被16D整除
    - 段基址的存放：实际地址的高16位存放在CPU段寄存器中

注意：CS的值不用也不允许由传送指令赋值，而是由段间调用、段间返回、中断指令等由系统改变的



1. 存储器的分段结构

- 寻址：8086CPU寄存器16 bits，地址数据20 bits，无法直接存储
  - **存储器分段技术**：
    - 偏移地址：单元所在位置距离其所在段段首单元的距离，段首单元的偏移地址为0000H，后续的单元顺次增1。



1. 存储器的分段结构

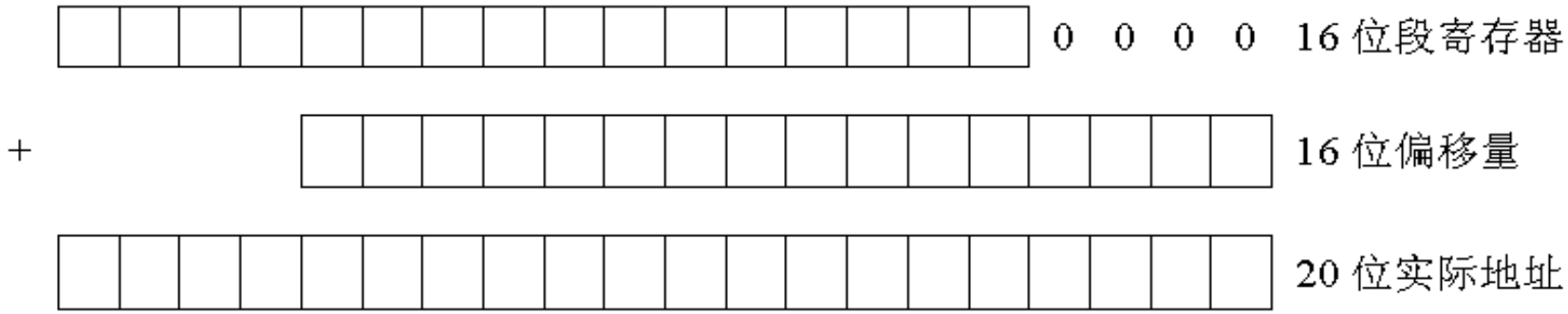
- 寻址：8086CPU寄存器16 bits，地址数据20 bits，无法直接存储
  - **存储器分段技术**：
    - 偏移地址：单元所在位置距离其所在段段首单元的距离，段首单元的偏移地址为0000H，后续的单元顺次增1。
    - **偏移地址存放**：
      - 指针变址寄存器SI，DI，BP，SP存放的是在某一段内寻址的单元的偏移地址。
      - 其中SI和DI存放的是数据段内某单元的偏移地址
      - 而BP和SP存放的则是堆栈段内某单元的偏移地址。
      - 指令指针IP用以存放下一条要执行的指令在当前代码段内的偏移地址。

2. 实际地址的产生

- 逻辑地址：两部分组成，段基址和偏移地址
- 实际地址：内存单元的物理地址

2. 实际地址的产生

- 逻辑地址：两部分组成，段基址和偏移地址
- 实际地址：内存单元的物理地址
- 实际地址 = 段基址左移4位，加偏移地址



- 例子：
  - 代码段，CS = 9482，IP = 2350。实际地址？

2. 实际地址的产生

- 例子：
  - 代码段，CS = 9482，IP = 2350。实际地址？

1001 0100 1000 0010 0000　(CS << 4)

0010 0011 0101 0000　(IP)

1001 0110 1011 0111 0000

2. 实际地址的产生

- 例子：
  - 代码段，CS = 9482，IP = 2350。实际地址？

1001 0100 1000 0010 0000　(CS << 4)

0010 0011 0101 0000　(IP)

1001 0110 1011 0111 0000

注意：一个实际地址可以对应多个逻辑地址。

第三章 微型计算机的结构

三. 寻址方式

1. 操作数的种类
2. 固定寻址
3. 立即寻址
4. 寄存器直接寻址
5. 存储器寻址（存储器直接寻址，寄存器间接寻址，基址寻址，变址寻址，基变址寻址）
6. 数据串寻址（暂不讲）

1. 操作数（Operands）的种类

1） 立即操作数：指令要操作的数据在指令代码中，MOV AL, 10H

2） 存储器操作数：指令要操作的数据在存储器(内存)中，MOV AL, [1234H]

3） 寄存器操作数：指令要操作的数据在CPU的寄存器中，MOV AL, BL

4） I/O端口操作数：指令要操作的数据来自或送到I/O， IN AL，20H

2. 固定寻址

- 定义：指令要操作的数据在指令中并没有明确给出，但**隐含**在指令中
- 例子： MUL BL ; AL*BL → AX

在该指令中，AL和AX并未给出。

3. 立即寻址

- 定义：指令要操作的数据包含在指令码中
- 例子： MOV AX, 1234H

4. 寄存器直接寻址

- 定义：指令(码)中给出的寄存器的名字(编号), 要操作的数据在该寄存中
- 例子：

```
INC  CX
INC  DX
INC  BX
INC  SP
INC  BP
```

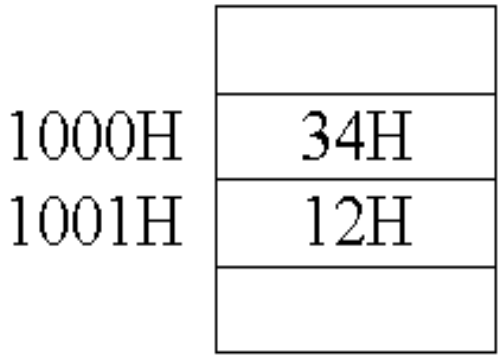
4. 存储器寻址

- 定义：要寻址的数据位于存储器(内存)中，在指令中是直接或间接的给出的存储器操作数的地址
- **存储器寻址包括以下几类：**
  1) 存储器直接寻址
  2) 寄存器间接寻址
  3) 基址寻址
  4) 变址寻址
  5) 基变址寻址

4. 存储器寻址

- 存储器寻址包括以下几类：
  1) **存储器直接寻址**：在存储器直接寻址中，指令直接给出的是操作数在内存中存放的地址

```
MOV  AL, [1000H]
MOV  BX, [1000H]
```



4. 存储器寻址

- 存储器寻址包括以下几类：
  1) 存储器直接寻址
  2) **寄存器间接寻址**：在寄存器间接寻址中，操作数位于内存中，操作数的地址位于某个寄存器中，在指令(码)中给出的是该寄存器的名字(编号)。

```
MOV  AL, [BX]
MOV  BL, [SI]
```

Case1: BX=1001H, SI = 1000H

Case2: BX=1002H, SI = 1000H

1000H	34H
1001H	12H

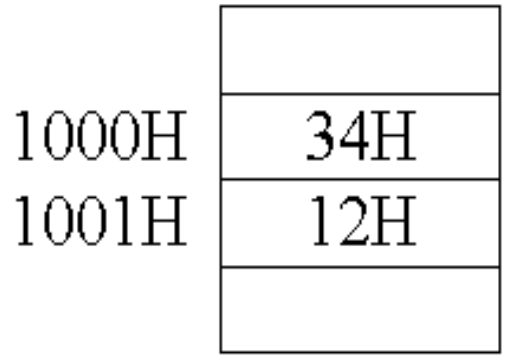
4. 存储器寻址

- 存储器寻址包括以下几类：
  1) 存储器直接寻址
  2) **寄存器间接寻址**：在寄存器间接寻址中，操作数位于内存中，操作数的地址位于某个寄存器中，在指令(码)中给出的是该寄存器的名字(编号)。
  注意：可以用于寄存器间接寻址的寄存器有BX, SI, DI

```
MOV  AL, [BX]
MOV  BL, [SI]
```

Case1: BX=1001H, SI = 1000H

Case2: BX=1002H, SI = 1000H

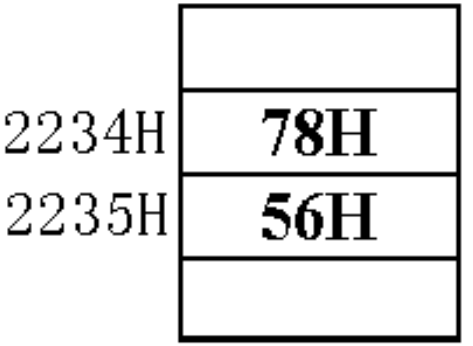


4. 存储器寻址

- 存储器寻址包括以下几类：
  1) 存储器直接寻址
  2) 寄存器间接寻址
  3) **基址寻址**：在基址寻址中，操作数位于位于内存中，操作数的地址由基址寄存器BX或BP与一个位移量相加给出，在指令(码)中给出的是该基址寄存器的名字(编号)及位移量

MOV AL, [BX+1234H]

BX=1000H



4. 存储器寻址

- 存储器寻址包括以下几类：
  1）存储器直接寻址
  2）寄存器间接寻址
  3）**基址寻址**：在基址寻址中，操作数位于位于内存中，操作数的地址由基址寄存器BX或BP与一个位移量相加给出，在指令(码)中给出的是该基址寄存器的名字(编号)及位移量

  注意：基址寻址的格式为 [BX+位移量] 或 [BP+位移量]
  位移量范围为补码表示的16位（ –32768~+32767 ）

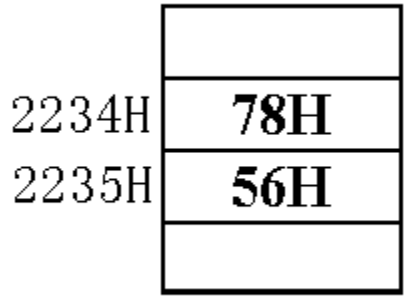
MOV AL, [BX+100H]
MOV AL, 100H[BX]

4. 存储器寻址

- 存储器寻址包括以下几类：
  1) 存储器直接寻址
  2) 寄存器间接寻址
  3) 基址寻址
  4) **变址寻址**：在变址寻址中，操作数位于位于内存中，操作数的地址由变址寄存器SI或DI与一个位移量相加给出，在指令(码)中给出的是该变址寄存器的名字(编号)及位移量

MOV AL, [SI+1234H]

SI = 1000H



4. 存储器寻址

- 存储器寻址包括以下几类：
  1）存储器直接寻址
  2）寄存器间接寻址
  3）基址寻址
  4）变址寻址：在变址寻址中，操作数位于位于内存中，操作数的地址由变址寄存器SI或DI与一个位移量相加给出，在指令(码)中给出的是该变址寄存器的名字(编号)及位移量

  注意：变址寻址的格式为[SI+位移量] 或 [DI+位移量]
  位移量范围为补码表示的16位（ –32768~+32767 ）

MOV AL, [SI+100H]
MOV AL, 100H[DI]

4. 存储器寻址

- 存储器寻址包括以下几类：
  1）存储器直接寻址
  2）寄存器间接寻址
  3）基址寻址
  4）变址寻址
  5）**基变址寻址**：在基变址寻址中，操作数位于位于内存中，操作数的地址由基址寄存器BX或BP与变址寄存器SI或DI及一个位移量相加给出，在指令(码)中给出的是寄存器的名字(编号)及位移量

MOV AL, [BX+SI+1234H]

BX = 1000H，SI = 2000H

4. 存储器寻址

- 存储器寻址包括以下几类：
  1）存储器直接寻址
  2）寄存器间接寻址
  3）基址寻址
  4）变址寻址
  5）基变址寻址：在基变址寻址中，操作数位于位于内存中，操作数的地址由基址寄存器BX或BP与变址寄存器SI或DI及一个位移量相加给出，在指令(码)中给出的是寄存器的名字(编号)及位移量

  注意：基变址寻址的格式为 [BX+SI+位移量]［BX+DI+位移量］[BP+SI+位移量]［BP+DI+位移量] [BX+SI]［BX+DI]［BP+SI］［BP+DI]

  错误写法：[BX+BP]［SI+DI]

  位移量范围为补码表示的16位（ –32768~+32767 ）

4. 存储器寻址

- 存储器寻址方式中的段地址
  - 在存储器寻址方式中只给出了偏移地址，其段地址是隐含的，一般情况下，是DS，只有特殊情况下是SS
  - 特殊情况：在基址寻址和基变址寻址方式下，基址寄存器是BP.
  - 例子：假定 DS=1000H, SS=2000H, BP=0100H, BX=0100H, 如下指令在执行完后的结果分别是什么？

MOV AX, [BX+100H]

MOV AX, [BP+100H]

1000:0200H	34H
0201H	12H

2000:0200H	78H
:0201H	56H

3. 段更换与段跨越

- **段更换**：当要操作的数据不在隐含段中时，就需要段更换或段跨越。要寻址的数据在2000H段的0100H单元，而目前没有一个段寄存的值是2000H，就需要将2000H装入某个段寄存器，如DS，这就是段更换

```
MOV   AX, 2000H
MOV   DS, AX
MOV   BX, 0100H
MOV   AL, [BX]
```

3. 段更换与段跨越

- **段跨越**：当要操作的数据不在隐含段中时，就需要段更换或段跨越.要寻址的数据在2000H段的0100H单元,而目前CS段寄存的值是2000H,可在寻址操作数加一段跨越前缀,如CS:

```
MOV    BX, 0100H
MOV    AL, CS:[BX]
```

4. 有效地址的计算时间

寻址方式	书写形式	计算时间
存储器直接寻址	DISP	6
寄存器间接寻址	[BX], [SI], [DI]	5
基址寻址	[BX+DISP], [BP+DISP]	9
变址寻址	[SI+DISP], [DI+DISP]	9
基变址寻址 (无位移量)	[BP+DI], [BX+SI] [BP+SI], [BX+DI]	7 8
基变址寻址 (有位移量)	[BP+DI+DISP], [BX+SI+DISP] [BP+SI+DISP], [BX+DI+DISP]	11 12

5. 指令系统

共分为14类92种指令：

(1)数据传送　　(8)循环控制

(2)算术运算　　(9)调用与返回

(3)逻辑运算　　(10)BCD调整

(4)移位　　(11)输入输出

(5)标志位操作　　(12)中断处理

(6)转移　　(13)外同步

(7)数据串操作　　(14)空操作