



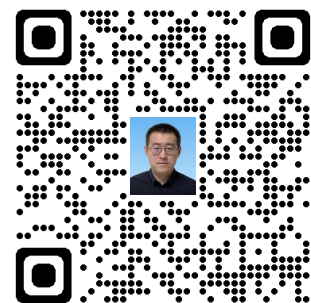
東北大學
Northeastern University

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. 寄存器再学习

四. 寻址方式

五. 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) — 回顾

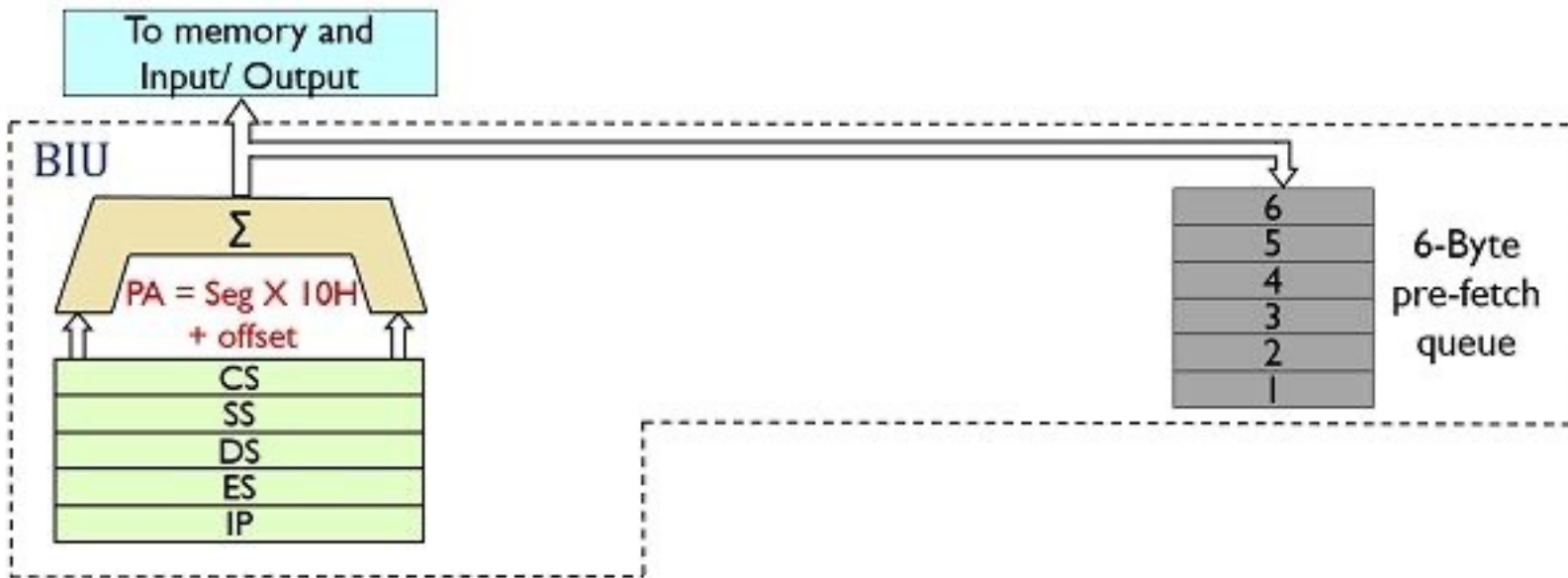
▶ 1. 8086/8088 CPU的结构

**8086 Inctruction Cycle
(Animated)**

B3 06	MOV BL, 06H
B1 05	MOV CL, 05H
00 CB	ADD BL, CL
内存	

一. 微处理器的结构(8086/8088) — 回顾

▶ 1. 8086/8088 CPU的结构

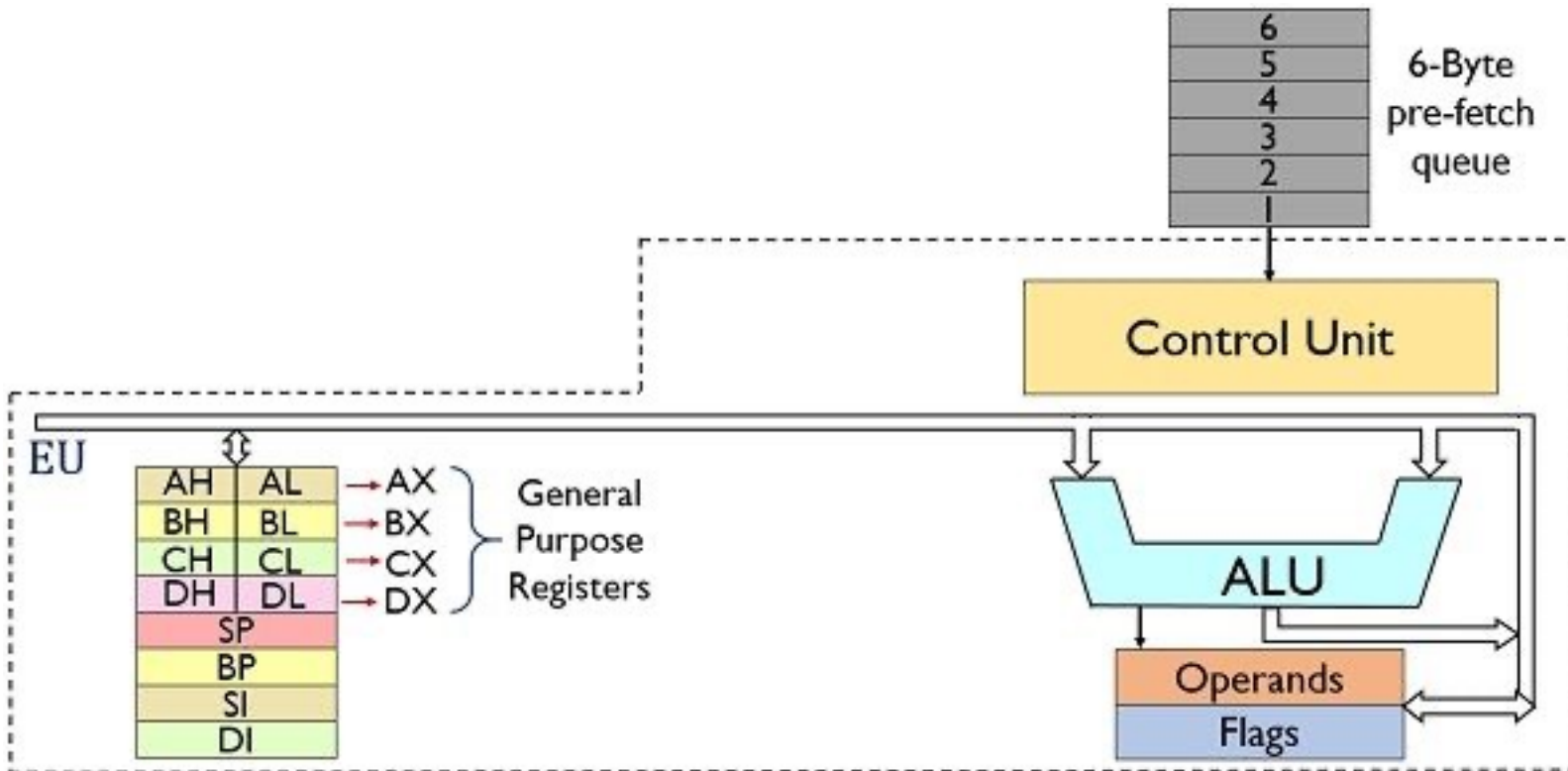


BIU:

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

一. 微处理器的结构(8086/8088) — 回顾

▶ 1. 8086/8088 CPU的结构

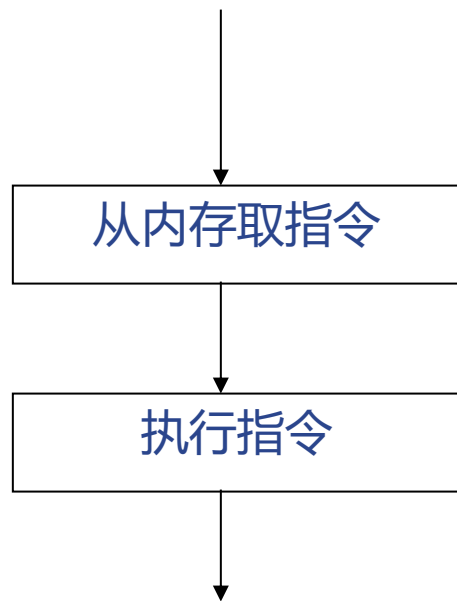


EU:

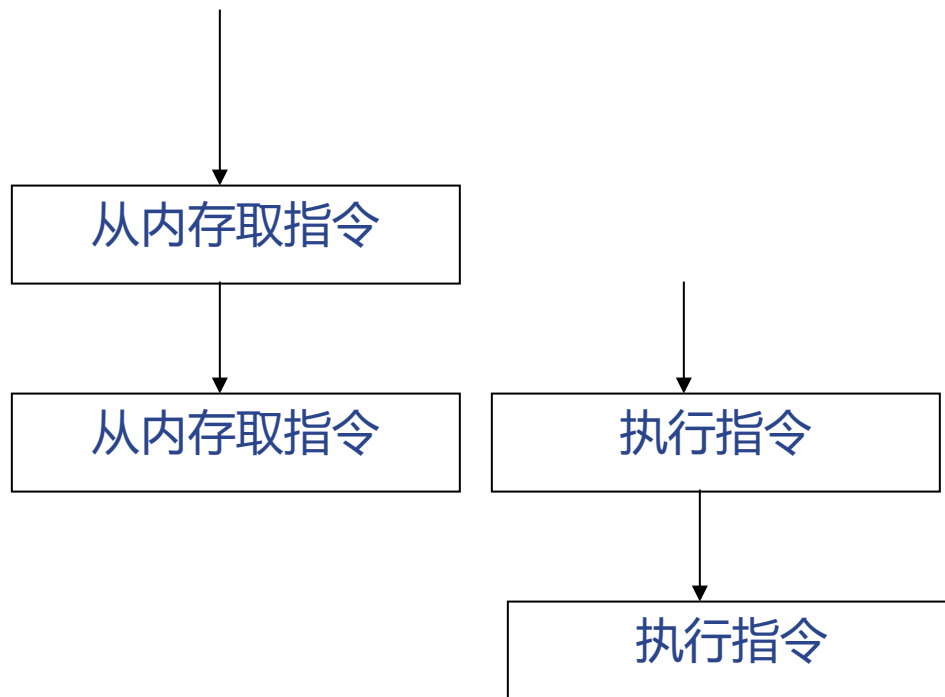
1. EU is responsible for the execution of instructions
2. It tells BIU from where to fetch instruction and data

一. 微处理器的结构(8086/8088) — 回顾

▶ 流水线 (pipeline)



传统CPU执行指令的过程

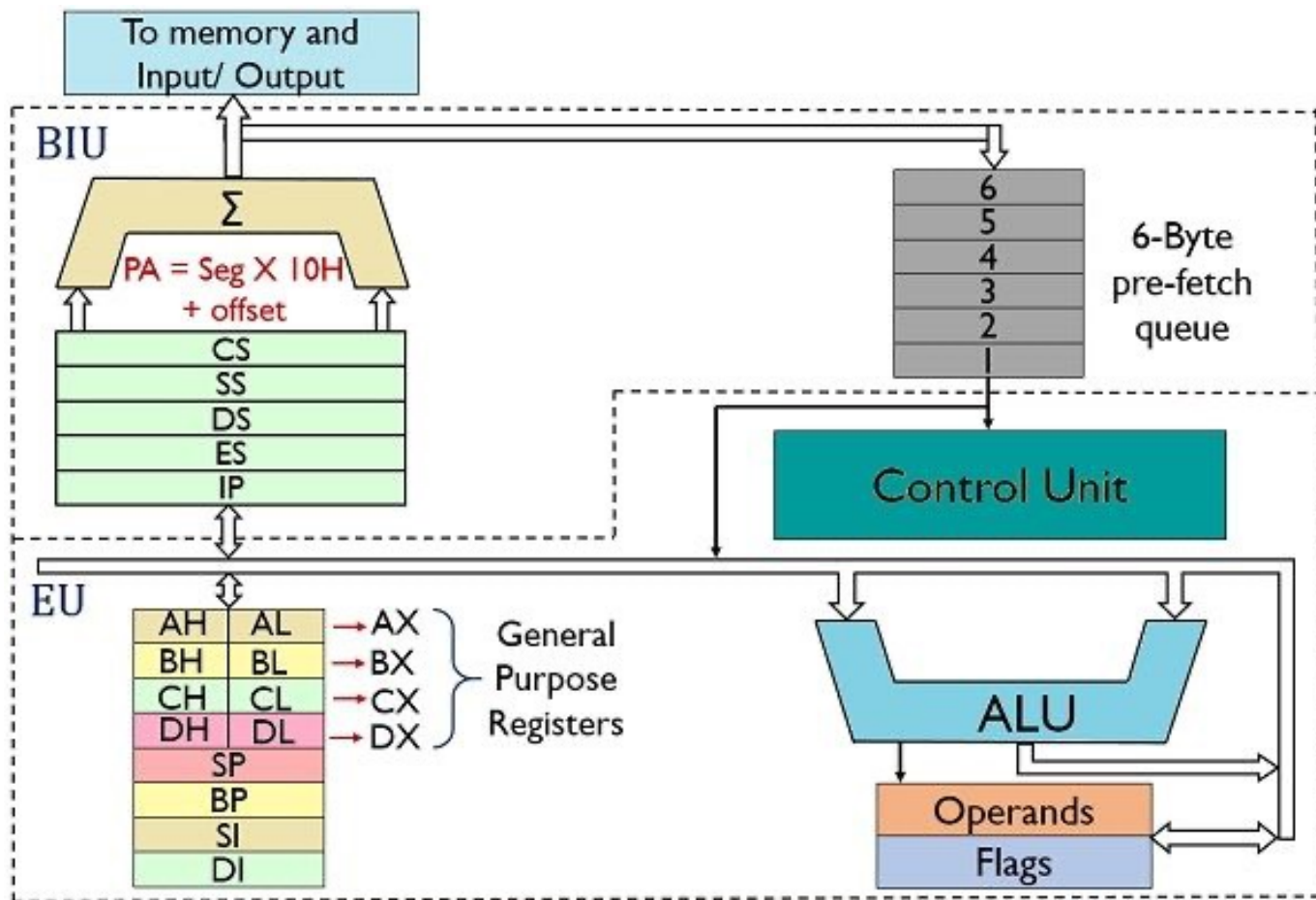


现代CPU执行指令的过程

一. 微处理器的结构(8086/8088) — 回顾

▶ 2. 8086/8088CPU的寄存器 (Register)

- 通用寄存器 (4个)
- 标志寄存器 (1个)
- 指针和变址寄存器 (5个)
- 段寄存器 (4个)



一. 微处理器的结构(8086/8088)

▶ 2. 8086/8088CPU的寄存器（四种类型）

- 通用寄存器：AX, BX, CX, DX (16-bit)
 - 主要用于数据传送和数据运算

一. 微处理器的结构(8086/8088)

▶ 2. 8086/8088CPU的寄存器（四种类型）

- 通用寄存器：AX, BX, CX, DX (16-bit)
 - 主要用于数据传送和数据运算

指令名称	格式	操作	说明	指令分类
MOV	MOV dest, src	dest ← src	move (传送)	数据传送
ADD	ADD dest, src	dest ← dest+src	addition (加法)	算数运算
SUB	SUB dest, src	dest ← dest-src	subtract (减法)	算数运算

一. 微处理器的结构(8086/8088)

▶ 2. 8086/8088CPU的寄存器（四种类型）

- 通用寄存器：AX, BX, CX, DX (16-bit)

➤ 主要用于数据传送和数据运算

- 例：（原AX=0000H，BX=0000H）

行号	程序指令	执行后AX中数据	执行后BX中数据
1	MOV AX, 4E20H		
2	ADD AX, 1406H		
3	MOV BX, 2000H		
4	ADD AX, BX		
5	MOV BX, AX		
6	ADD AX, BX		

一. 微处理器的结构(8086/8088)

▶ 2. 8086/8088CPU的寄存器（四种类型）

- 通用寄存器：AX, BX, CX, DX (16-bit)

➤ 主要用于数据传送和数据运算

- 例：（原AX=0000H，BX=0000H）

行号	程序指令	执行后AX中数据	执行后BX中数据
1	MOV AX, 4E20H	4E20H	0000H
2	ADD AX, 1406H		
3	MOV BX, 2000H		
4	ADD AX, BX		
5	MOV BX, AX		
6	ADD AX, BX		

一. 微处理器的结构(8086/8088)

▶ 2. 8086/8088CPU的寄存器（四种类型）

- 通用寄存器：AX, BX, CX, DX (16-bit)

➤ 主要用于数据传送和数据运算

- 例：（原AX=0000H，BX=0000H）

行号	程序指令	执行后AX中数据	执行后BX中数据
1	MOV AX, 4E20H	4E20H	0000H
2	ADD AX, 1406H	6226H	0000H
3	MOV BX, 2000H	6226H	2000H
4	ADD AX, BX		
5	MOV BX, AX		
6	ADD AX, BX		

一. 微处理器的结构(8086/8088)

▶ 2. 8086/8088CPU的寄存器（四种类型）

- 通用寄存器：AX, BX, CX, DX (16-bit)

➤ 主要用于数据传送和数据运算

- 例：（原AX=0000H，BX=0000H）

行号	程序指令	执行后AX中数据	执行后BX中数据	
1	MOV AX, 4E20H	4E20H	0000H	8226H
2	ADD AX, 1406H	6226H	0000H	+ 8226H
3	MOV BX, 2000H	6226H	2000H	1044CH
4	ADD AX, BX	8226H	2000H	
5	MOV BX, AX	8226H	8226H	
6	ADD AX, BX	?	8226H	

一. 微处理器的结构(8086/8088)

▶ 2. 8086/8088CPU的寄存器（四种类型）

- 通用寄存器：AX, BX, CX, DX (16-bit)

- ▶ 主要用于数据传送和数据运算

- 例：（原AX=0000H，BX=0000H）

行号	程序指令	执行后AX中数据	执行后BX中数据	
1	MOV AX, 4E20H	4E20H	0000H	8226H
2	ADD AX, 1406H	6226H	0000H	+ 8226H
3	MOV BX, 2000H	6226H	2000H	<u>1044CH</u>
4	ADD AX, BX	8226H	2000H	
5	MOV BX, AX	8226H	8226H	
6	ADD AX, BX	044CH	8226H	

一. 微处理器的结构(8086/8088)

▶ 2. 8086/8088CPU的寄存器（四种类型）

- **通用寄存器：AX, BX, CX, DX (16-bit)**

- ▶ 主要用于数据传送和数据运算

- ▶ 每个通用寄存器可以分为两个8位寄存器，单独使用

- AH, AL
- BH, BL
- CH, CL
- DH, DL

15	8	7	0	
AH		AL		AX
BH		BL		BX
CH		CL		CX
DH		DL		DX
High-8		Low-8		

一. 微处理器的结构(8086/8088)

▶ 2. 8086/8088CPU的寄存器（四种类型）

- **通用寄存器：AX, BX, CX, DX (16-bit)**
 - 主要用于数据传送和数据运算
 - 每个通用寄存器可以分为两个8位寄存器，单独使用
 - 在进行数据传送或运算时，两个**操作对象的位数**应当保持一致

行号	程序指令
1	MOV AX, BX
2	MOV BX, CX
3	MOV AX, 18H
4	MOV AL, 18H
5	ADD BX, AX
6	ADD AX, 20000

行号	程序指令
1	MOV AX, BL
2	MOV BH, AX
3	MOV AL, 20000
4	ADD AL, 350H

一. 微处理器的结构(8086/8088)

▶ 2. 8086/8088CPU的寄存器（四种类型）

- **通用寄存器：AX, BX, CX, DX (16-bit)**
 - 主要用于数据传送和数据运算
 - 每个通用寄存器可以分为两个8位寄存器，单独使用
 - 在进行数据传送或运算时，两个**操作对象的位数**应当保持一致

行号	程序指令
1	MOV AX, BX
2	MOV BX, CX
3	MOV AX, 18H
4	MOV AL, 18H
5	ADD BX, AX
6	ADD AX, 20000



行号	程序指令
1	MOV AX, BL
2	MOV BH, AX
3	MOV AL, 20000
4	ADD AL, 350H



一. 微处理器的结构(8086/8088)

▶ 2. 8086/8088CPU的寄存器（四种类型）

- **通用寄存器：AX, BX, CX, DX (16-bit)**
 - 每个通用寄存器，一般用途时可以互换

一. 微处理器的结构(8086/8088)

▶ 2. 8086/8088CPU的寄存器（四种类型）

- **通用寄存器：AX, BX, CX, DX (16-bit)**
 - ▶ 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)：累加器（乘除法指令），数据寄存器等
 - BX (Base)：基址寄存器（基址寻址）等
 - CX (Count)：计数器寄存器（循环、移位等指令中作为次数计数器）等
 - DX (Data)：数据寄存器（I/O地址存储）等

一. 微处理器的结构(8086/8088)

▶ 2. 8086/8088CPU的寄存器（四种类型）

- **通用寄存器：AX, BX, CX, DX (16-bit)**
 - 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)：**累加器（乘法指令）**，数据寄存器等

指令名称	格式	操作	说明	指令分类
MUL	MUL src	AX ← AL * src (字节乘法) DX, AX ← AX * src (字乘法)	multiply, unsigned (无符号乘法)	算术运算

一. 微处理器的结构(8086/8088)

▶ 2. 8086/8088CPU的寄存器（四种类型）

- **通用寄存器：AX, BX, CX, DX (16-bit)**

- ▶ 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)：**累加器（乘法指令）**，数据寄存器等

- 例：100*10 = 1000（原AX=0000H，BX=0000H）

行号	程序指令	AH (after)	AL (after)	BL (after)
1	MOV AL, 100			
2	MOV BL, 10			
3	MUL BL			

指令名称	格式	操作	说明	指令分类
MUL	MUL src	AX ← AL * src (字节乘法) DX, AX ← AX * src (字乘法)	multiply, unsigned (无符号乘法)	算术运算

一. 微处理器的结构(8086/8088)

▶ 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

一. 微处理器的结构(8086/8088)

▶ 2. 8086/8088CPU的寄存器

- 标志寄存器* (F) – ZF

➤ 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

– 操作结果为0，ZF = 1

– 操作结果不为0，ZF = 0

$$\begin{array}{r} 1111\ 1111 \\ 0000\ 0001 \\ + \\ 0000\ 0000 \\ \hline \end{array}$$

进位1 ←

操作结果为0, ZF=1

一. 微处理器的结构(8086/8088)

▶ 2. 8086/8088CPU的寄存器

- 标志寄存器* (F) – PF

➤ 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“1”的个数，偶数时PF = 1
- 奇数个“1”，PF = 0

$$\begin{array}{r} \text{AND} \quad 1000\ 0101\ 1010\ 1100 \\ \quad \quad 1001\ 1101\ 1001\ 0111 \\ \hline 1000\ 0101\ 1000\ 0100 \end{array}$$

一. 微处理器的结构(8086/8088)

▶ 2. 8086/8088CPU的寄存器

- 标志寄存器* (F) – SF

➤ 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)，如果为负，SF = 1
- 如果非负，SF = 0

$$\begin{array}{r} 1000\ 0101\ 1010\ 1100 \\ \text{AND } 1001\ 1101\ 1001\ 0111 \\ \hline 1000\ 0101\ 1000\ 0100 \end{array}$$

结果: SF=1

一. 微处理器的结构(8086/8088)

▶ 2. 8086/8088CPU的寄存器

- 标志寄存器* (F) – SF

- 计算机中通常用补码来表示有符号数据，例如：

- 0000 0001B，可以看做是无符号的1，或者有符号的 +1
 - 1000 0001B，可以看做是无符号的129，或者有符号的 -127

一. 微处理器的结构(8086/8088)

▶ 2. 8086/8088CPU的寄存器

- 标志寄存器* (F) – SF

- ▶ 计算机中通常用补码来表示有符号数据，例如：
 - 0000 0001B，可以看做是无符号的1，或者有符号的 +1
 - 1000 0001B，可以看做是无符号的129，或者有符号的 -127
- ▶ 在补码表示中，仍用0表示正数，1表示负数。对于正数，其补码表示与其原码表示完全相同；**对于负数，符号位为1，其余各位按位取反加1**

源码→补码：-127 → 1111 1111B → 1000 0000B → 1000 0001B

一. 微处理器的结构(8086/8088)

▶ 2. 8086/8088CPU的寄存器

- 标志寄存器* (F) – SF

- 计算机中通常用补码来表示有符号数据，例如：
 - 0000 0001B，可以看做是无符号的1，或者有符号的 +1
 - 1000 0001B，可以看做是无符号的129，或者有符号的 -127
- 在补码表示中，仍用0表示正数，1表示负数。对于正数，其补码表示与其原码表示完全相同；**对于负数，符号位为1，其余各位按位取反加1**

源码→补码：-127 → 1111 1111B → 1000 0000B → 1000 0001B

补码→源码：**1000 0001B** → 1000 0000B → 1111 1111B → -127

一. 微处理器的结构(8086/8088)

▶ 2. 8086/8088CPU的寄存器

- **标志寄存器* (F) – SF**

- 计算机中通常用补码来表示有符号数据，例如：
 - 0000 0001B，可以看做是无符号的1，或者有符号的 +1
 - 1000 0001B，可以看做是无符号的129，或者有符号的 -127
- 对于同一个二进制数据，计算机可以将它当作无符号数据来运算，也可以当作有符号数据来运算

一. 微处理器的结构(8086/8088)

▶ 2. 8086/8088CPU的寄存器

- **标志寄存器* (F) – SF**

- ▶ 计算机中通常用补码来表示有符号数据，例如：
 - 0000 0001B，可以看做是无符号的1，或者有符号的 +1
 - 1000 0001B，可以看做是无符号的129，或者有符号的 -127
- ▶ 对于同一个二进制数据，计算机可以将它当作无符号数据来运算，也可以当作有符号数据来运算

行号	程序指令	AL(after)	SF(after)
1	MOV AL, 1000 0001B		
2	ADD AL, 1	1000 0010B	1
...

解释1：无符号加法时，结果为130

解释2：有符号加法时，结果为-126

一. 微处理器的结构(8086/8088)

▶ 2. 8086/8088CPU的寄存器

- **标志寄存器* (F) – SF**

- 计算机中通常用补码来表示有符号数据，例如：
 - 0000 0001B，可以看做是无符号的1，或者有符号的 +1
 - 1000 0001B，可以看做是无符号的129，或者有符号的 -127
- 对于同一个二进制数据，计算机可以将它当作无符号数据来运算，也可以当作有符号数据来运算

一. 微处理器的结构(8086/8088)

▶ 2. 8086/8088CPU的寄存器

- **标志寄存器* (F) – CF**

➤ 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
- 有进位/借位发生时，CF = 1
- 同SF一样，是一种记录，无符号
数运算时使用

一. 微处理器的结构(8086/8088)

▶ 2. 8086/8088CPU的寄存器

- 标志寄存器* (F) – CF

➤ 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
- 有进位/借位发生时，CF = 1
- 同SF一样，是一种记录，无符号数运算时使用

行号	程序指令	AL(after)	CF(after)
1	MOV AL, 98H	98H (1001 1000B)	未知
2	ADD AL, AL		
3	ADD AL, AL		

一. 微处理器的结构(8086/8088)

▶ 2. 8086/8088CPU的寄存器

- **标志寄存器* (F) – CF**

➤ 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
- 有进位/借位发生时，CF = 1
- 同SF一样，是一种记录，无符号数运算时使用

行号	程序指令	AL(after)	CF(after)
1	MOV AL, 97H		
2	SUB AL, 98H		
3	SUB AL, AL		

一. 微处理器的结构(8086/8088)

▶ 2. 8086/8088CPU的寄存器

- 标志寄存器* (F) – AF

➤ 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
- 有进位/借位发生时，CF = 1
- 同SF一样，是一种记录，无符号数运算时使用

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

$$\begin{array}{r} 0000\ 0000 \\ - 0000\ 0001 \\ \hline 1111\ 1111 \end{array}$$


一. 微处理器的结构(8086/8088)

▶ 2. 8086/8088CPU的寄存器

- **标志寄存器* (F) – OF**

➤ 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

- 如果结果超过目标所能容纳的范围，OF = 1
- 未超出所能容纳的范围，OF = 0

一. 微处理器的结构(8086/8088)

▶ 2. 8086/8088CPU的寄存器

- **标志寄存器* (F) – OF标志**

- 溢出：在进行**有符号数运算**的时候，如结果超过了机器所能表示的范围称为溢出。
 - 8位有符号数：-128 ~ 127
 - 16位有符号数：-32768~32767
- 注意，这里所讲的溢出，只是**对有符号数运算**而言

一. 微处理器的结构(8086/8088)

▶ 2. 8086/8088CPU的寄存器

- 标志寄存器* (F) – OF标志

- ▶ 溢出：在进行有符号数运算的时候，如结果超过了机器所能表示的范围称为溢出。
 - 8位有符号数：-128 ~ 127
 - 16位有符号数：-32768~32767
- ▶ 注意，这里所讲的溢出，只是对有符号数运算而言

行号	程序指令	AL (after)	OF (after)
1	MOV AL, F0H		
2	ADD AL, 88H		

转2进制 减1 数据位取反 转16进制 转10进制
F0H → 1111 0000B → 1110 1111B → 1001 0000B → -10H → -16

一. 微处理器的结构(8086/8088)

▶ 2. 8086/8088CPU的寄存器

- **标志寄存器* (F) – OF标志**

- ▶ 溢出：在进行**有符号数运算**的时候，如结果超过了机器所能表示的范围称为溢出。
 - 8位有符号数：-128 ~ 127
 - 16位有符号数：-32768~32767
- ▶ 注意，这里所讲的溢出，只是**对有符号数运算**而言

行号	程序指令	AL (after)	OF (after)
1	MOV AL, F0H (-16)		
2	ADD AL, 88H		

转2进制 减1 数据位取反 转16进制 转10进制
88H → 1000 1000B → 1000 0111B → 1111 1000B → -78H → -120

一. 微处理器的结构(8086/8088)

▶ 2. 8086/8088CPU的寄存器

- 标志寄存器* (F) – OF标志

- ▶ 溢出：在进行有符号数运算的时候，如结果超过了机器所能表示的范围称为溢出。
 - 8位有符号数：-128 ~ 127
 - 16位有符号数：-32768~32767
- ▶ 注意，这里所讲的溢出，只是对有符号数运算而言

行号	程序指令	AL (after)	OF (after)
1	MOV AL, F0H (-16)	F0H	未知
2	ADD AL, 88H (-120)	78H	1

一. 微处理器的结构(8086/8088)

▶ 2. 8086/8088CPU的寄存器

- **标志寄存器* (F) – OF标志和CF标志**

- CF是对无符号数运算有意义的标志位，记录无符号数运算是否产生了进位
- OF是对有符号数运算有意义的标志位，记录有符号数运算是否产生了溢出

行号	程序指令	CF (after)	OF (after)
1	MOV AL, 98	未知	未知
2	ADD AL, 99	0	1
3	MOV AL, F0H	0	1
4	ADD AL, 88H	1	1
5	MOV AL, F0H	1	1
6	ADD AL, 78H	1	0
...

一. 微处理器的结构(8086/8088)

▶ 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



一. 微处理器的结构(8086/8088)

▶ 2. 8086/8088CPU的寄存器

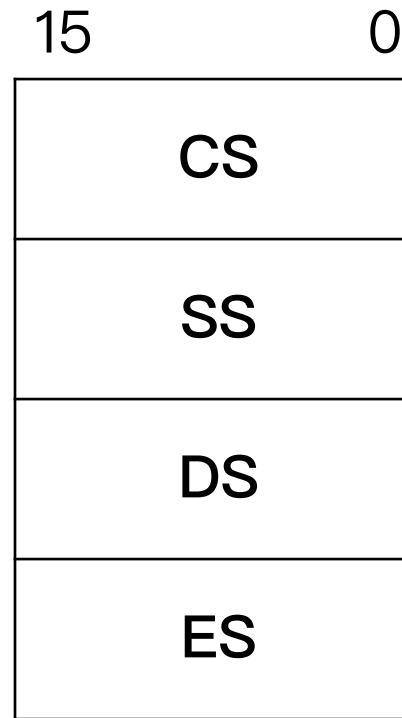
- **标志寄存器* (F) – 功能总结**
 - 用来存储相关指令的某些执行结果
 - 用来为CPU执行相关指令提供行为依据
 - 用来控制CPU的相关工作方式

一. 微处理器的结构(8086/8088)

▶ 2. 8086/8088CPU的寄存器 (Register)

- 段寄存器 (Segment Register) (CS, SS, DS, ES)

➤ “段”+“寄存器”



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)

一. 微处理器的结构(8086/8088)

▶ 2. 8086/8088CPU的寄存器 (Register)

- 段寄存器 (Segment Register) (CS, SS, DS, ES)

- ▶ “段”+“寄存器”

- 1) 段寄存器的作用？
- 2) 为什么要分“段”？
- 3) 段寄存器存储具体什么数据？

一. 微处理器的结构(8086/8088)

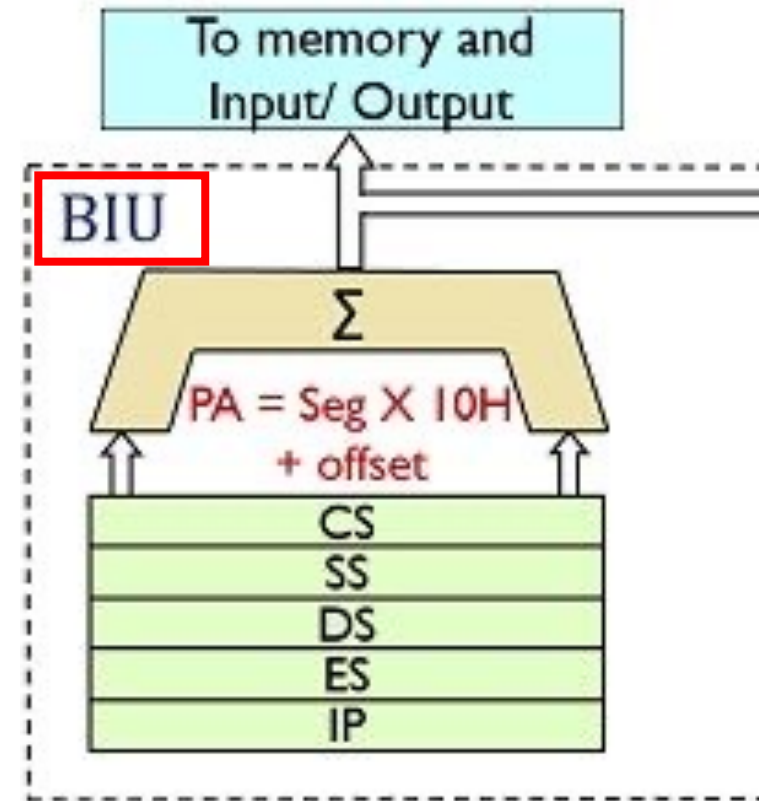
▶ 2. 8086/8088CPU的寄存器 (Register)

- 段寄存器 (Segment Register) (CS, SS, DS, ES)

- ▶ “段”+“寄存器”

- 1) 段寄存器的作用？

- 存储内存（存储器）地址相关信息



一. 微处理器的结构(8086/8088)

▶ 2. 8086/8088CPU的寄存器 (Register)

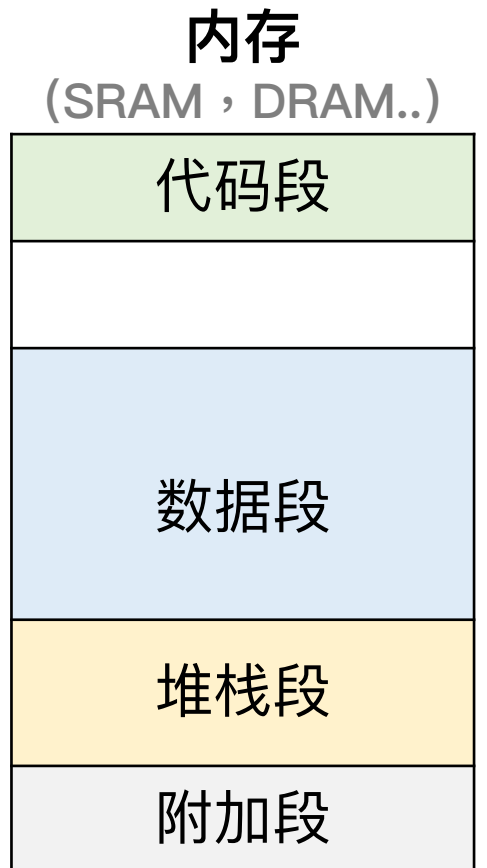
- 段寄存器 (Segment Register) (CS, SS, DS, ES)

- ▶ “段”+“寄存器”

- 1) 段寄存器的作用？

- 2) 为什么要分“段”？

- 区分内存中存储的内容，方便管理，快速访问



一. 微处理器的结构(8086/8088)

▶ 2. 8086/8088CPU的寄存器 (Register)

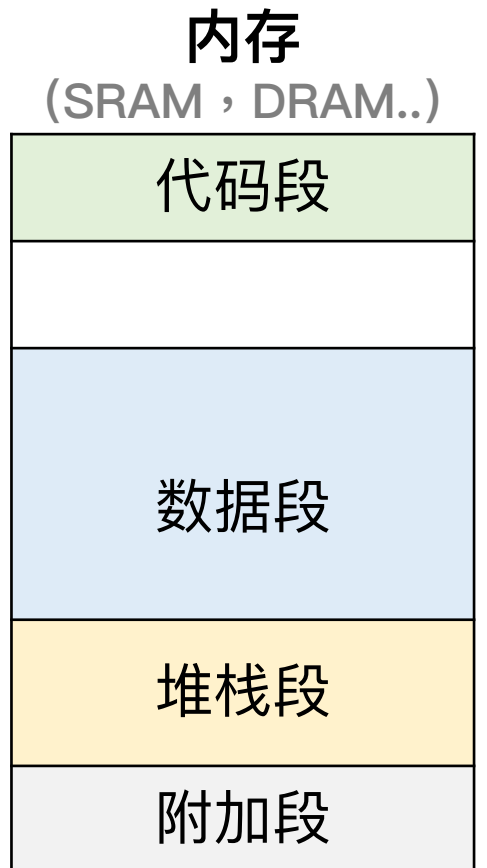
- 段寄存器 (Segment Register) (CS, SS, DS, ES)

- ▶ “段”+“寄存器”

- 1) 段寄存器的作用？

- 2) 为什么要分“段”？

- 区分内存中存储的内容，方便管理，快速访问
- 段的位置和大小是固定的吗？



一. 微处理器的结构(8086/8088)

▶ 2. 8086/8088CPU的寄存器 (Register)

- 段寄存器 (Segment Register) (CS, SS, DS, ES)

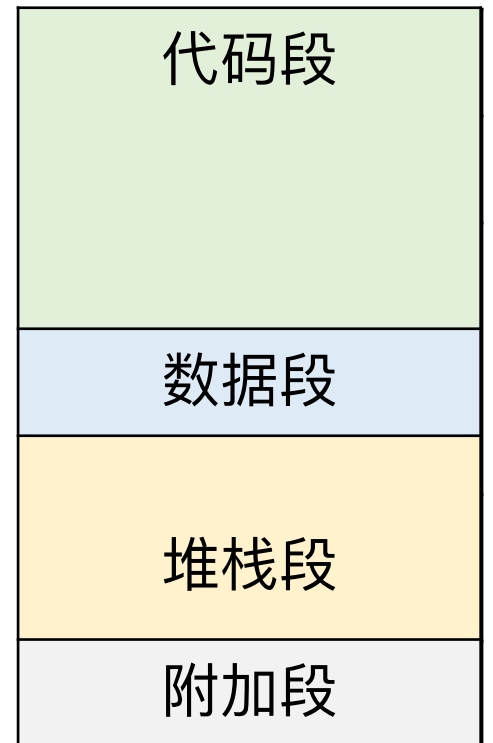
- ▶ “段”+“寄存器”

- 1) 段寄存器的作用？

- 2) 为什么要分“段”？

- 区分内存中存储的内容，方便管理，快速访问
- 段的位置和大小是固定的吗？
- 段是个逻辑概念，由程序员指定

内存
(SRAM, DRAM..)



一. 微处理器的结构(8086/8088)

▶ 2. 8086/8088CPU的寄存器 (Register)

- 段寄存器 (Segment Register) (CS, SS, DS, ES)

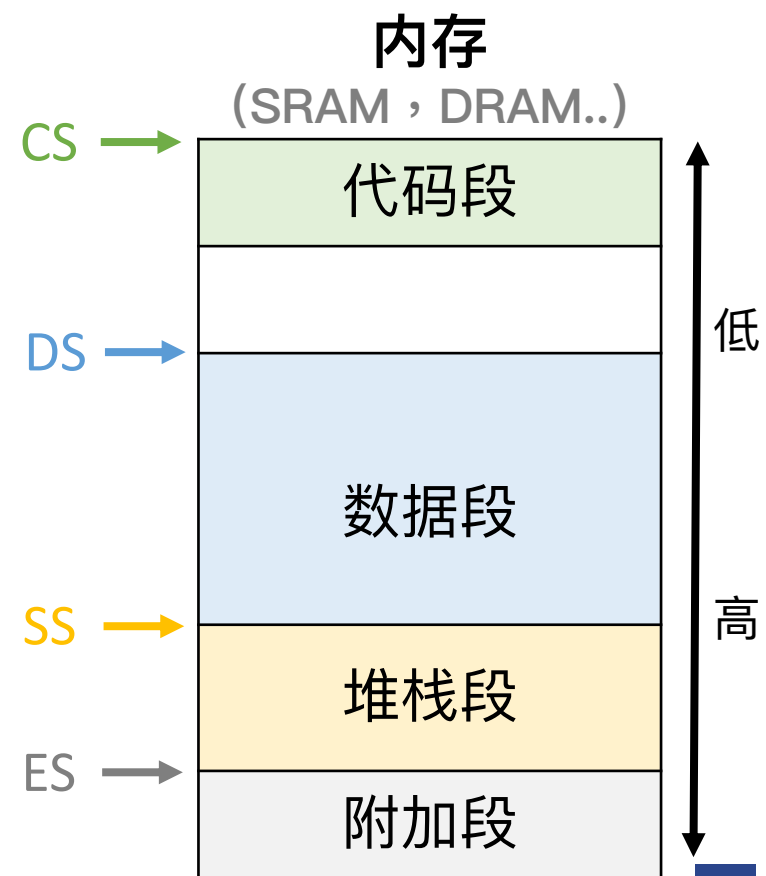
- ▶ “段”+“寄存器”

- 1) 段寄存器的作用？

- 2) 为什么要分“段”？

- 3) 段寄存器存储具体什么数据？

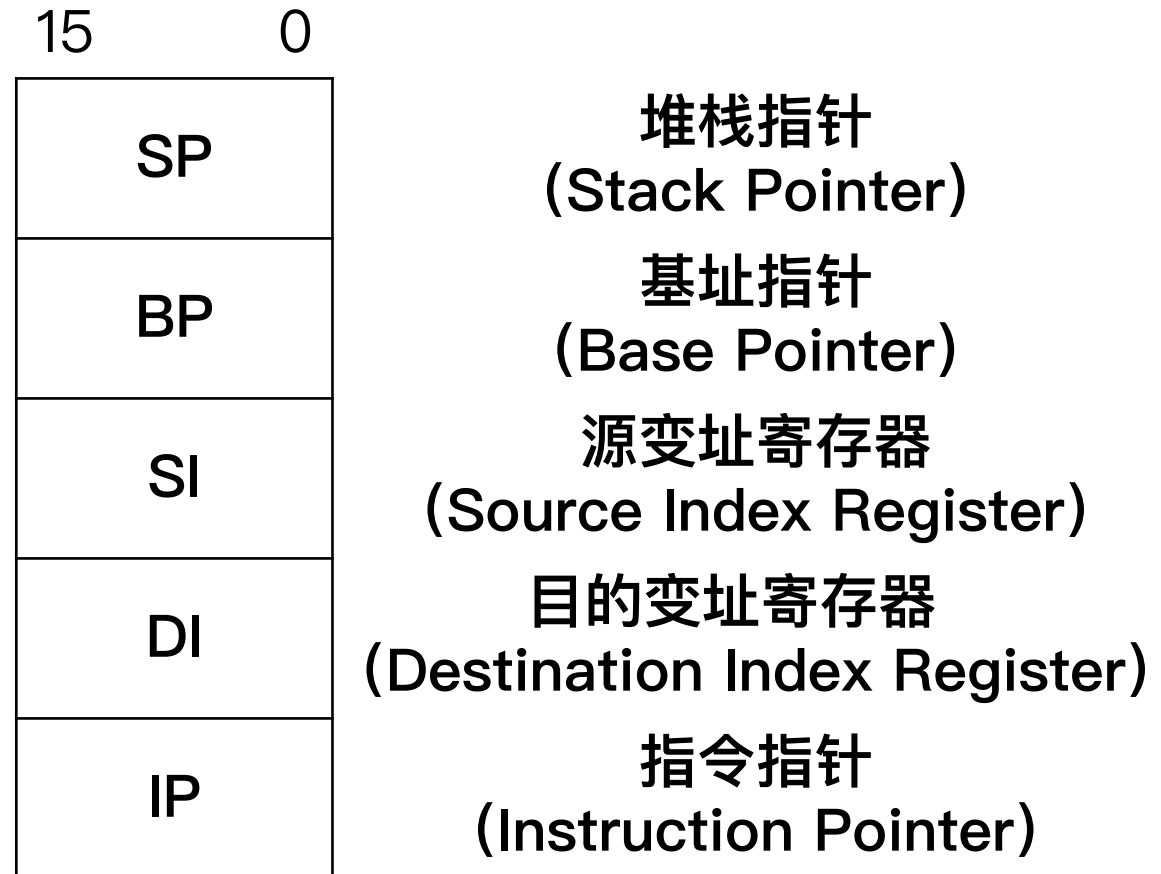
- CS：存储代码段基地址
- SS：存储堆栈段基地址
- DS：存储数据段基地址
- ES：存储附加段基地址



一. 微处理器的结构(8086/8088)

▶ 2. 8086/8088CPU的寄存器 (Register)

- 指针和变址寄存器 (特殊功能寄存器)



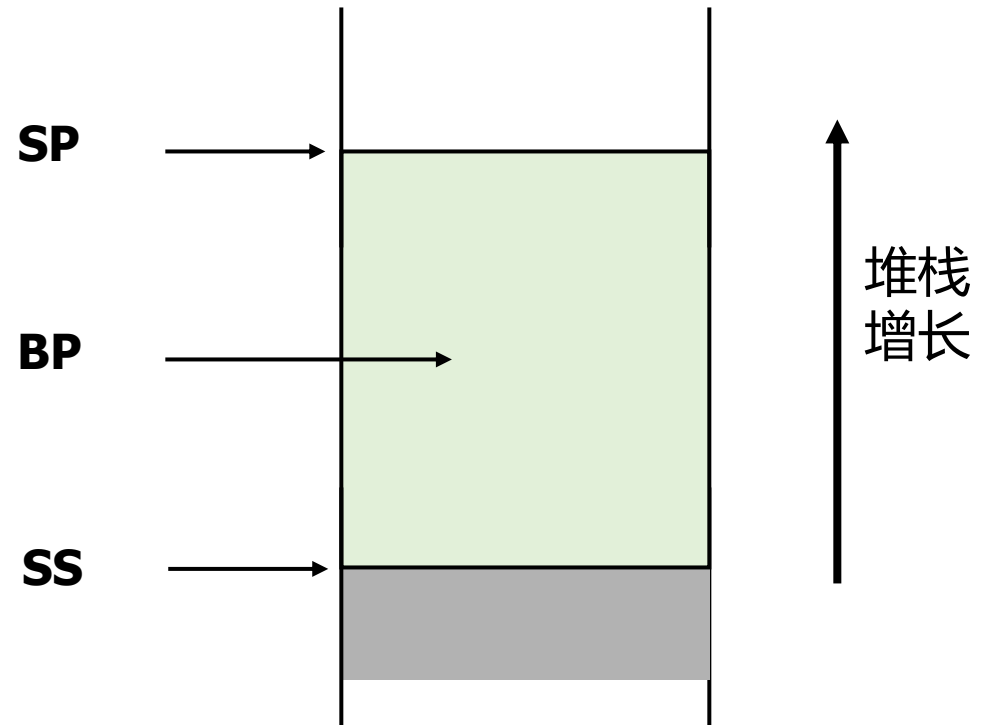
一. 微处理器的结构(8086/8088)

▶ 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



一. 微处理器的结构(8086/8088)

▶ 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

一. 微处理器的结构(8086/8088)

▶ 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

第三章 微型计算机的结构

二. 存储器(组织与结构)

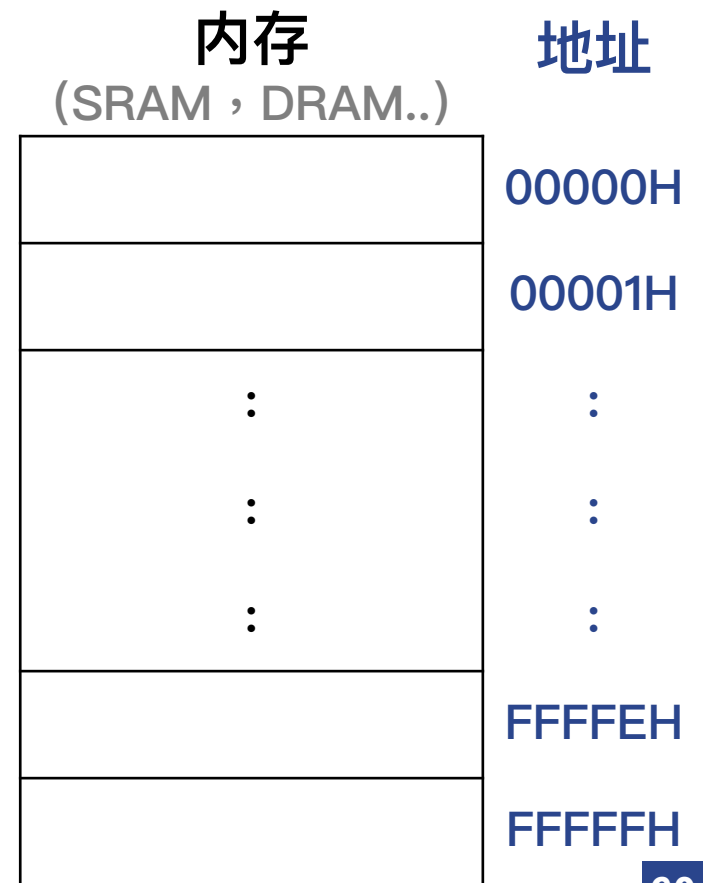
1. 存储器的分段结构
2. 实际地址的产生



二. 存储器(组织与结构)

▶ 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**



二. 存储器(组织与结构)

▶ 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？

内存 (SRAM, DRAM..)	地址
1 byte	00000H
1 byte	00001H
:	:
:	:
:	:
1 byte	FFFFEH
1 byte	FFFFFH

二. 存储器(组织与结构)

▶ 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)

内存 (SRAM, DRAM..)	地址
1 byte	00000H
1 byte	00001H
:	:
:	:
:	:
1 byte	FFFFEH
1 byte	FFFFFH

二. 存储器(组织与结构)

▶ 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

内存 (SRAM, DRAM..)	地址
1 byte	00000H
1 byte	00001H
:	:
:	:
:	:
1 byte	FFFFEH
1 byte	FFFFFH

二. 存储器(组织与结构)

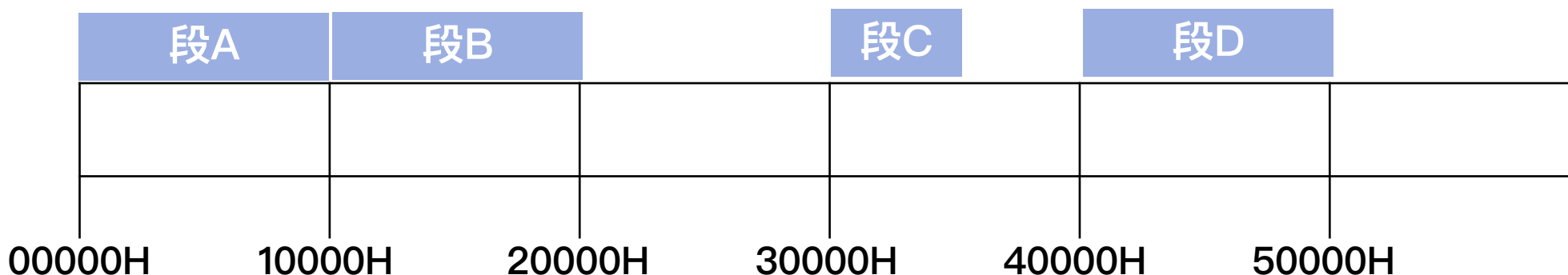
▶ 1. 存储器的分段结构

- 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 $\rightarrow 2^{16} = 64\text{KB}$
 - $(1/16) * 1\text{MB}$

二. 存储器(组织与结构)

▶ 1. 存储器的分段结构

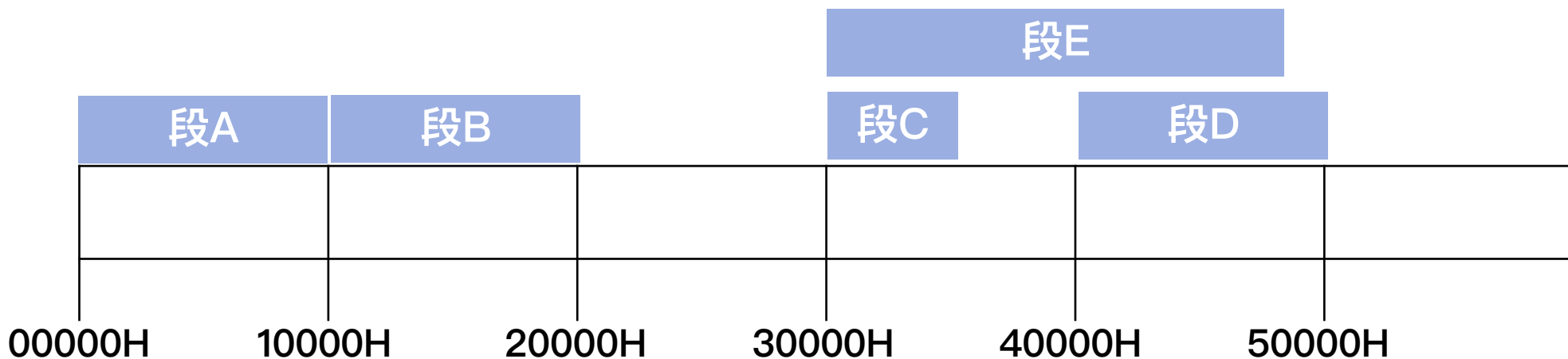
- 寻址：8086CPU寄存器16 bits，地址数据20 bits，无法直接存储
 - 存储器分段技术：
 - 将1M的地址空间，逻辑上分成多个段
 - 每个段最大长度 $\leq 64\text{KB}$



二. 存储器(组织与结构)

▶ 1. 存储器的分段结构

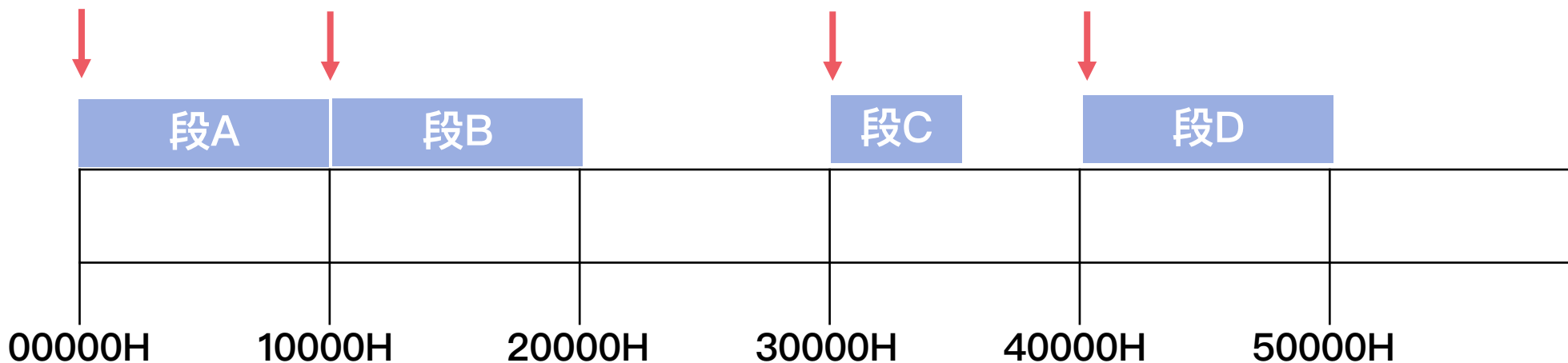
- 寻址：8086CPU寄存器16 bits，地址数据20 bits，无法直接存储
 - 存储器分段技术：
 - 段与段之间可以部分/全部重叠



二. 存储器(组织与结构)

▶ 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

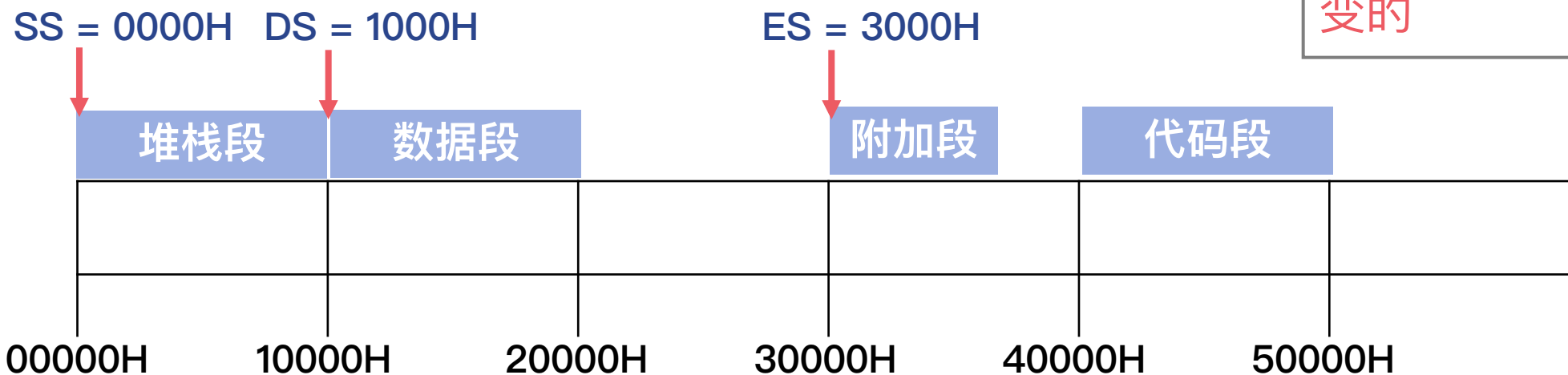


二. 存储器(组织与结构)

▶ 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

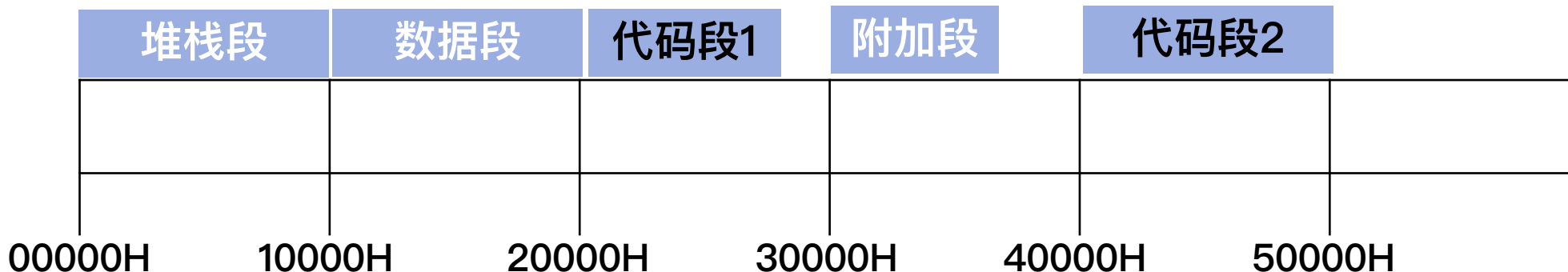


二. 存储器(组织与结构)

▶ 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

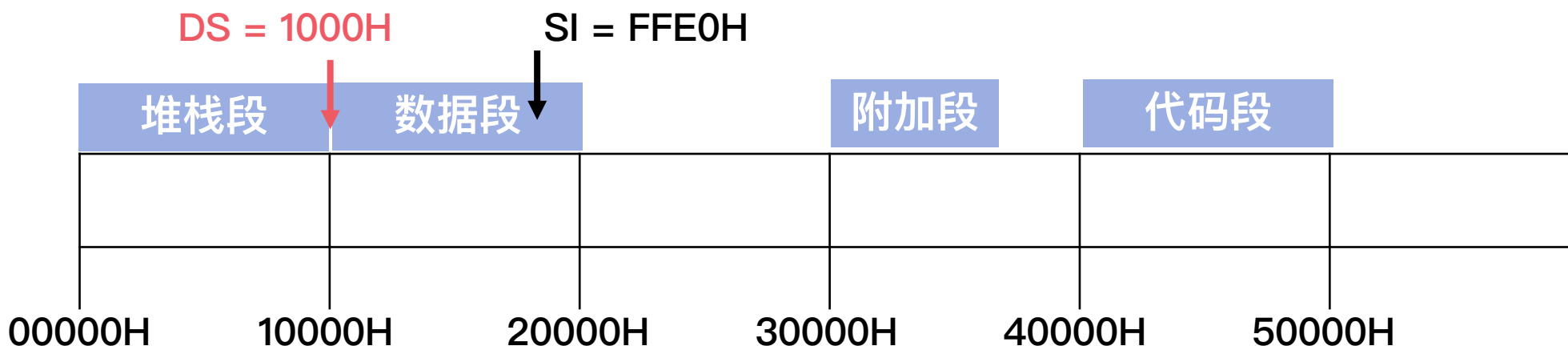
注意：CS的值不用也不允许由传送指令赋值，而是由段间调用、段间返回、中断指令等由系统改变的



二. 存储器(组织与结构)

▶ 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。



二. 存储器(组织与结构)

▶ 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。

二. 存储器(组织与结构)

▶ 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

二. 存储器(组织与结构)

▶ 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 = 9482H，IP = 2350H。实际地址？

二. 存储器(组织与结构)

▶ 2. 实际地址的产生

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

1001 0100 1000 0010 0000 (CS << 4)

0010 0011 0101 0000 (IP)

1001 0110 1011 0111 0000

二. 存储器(组织与结构)

▶ 2. 实际地址的产生

- 例子：
 - ▶ 代码段，CS = 9482H，IP = 2350H。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。

二. 存储器(组织与结构)

▶ 2. 实际地址的产生

- 实际地址的表示：
 - 例：数据在21F60H中
 - 1) 数据在2000:1F60内存单元中
 - 2) 数据在2000H段中的1F60H单元中

二. 存储器(组织与结构)

▶ 2. 实际地址的产生

- 实际地址的表示：

▶ 例：数据在21F60H中

- 1) 数据在2000:1F60内存单元中
- 2) 数据在2000H段中的1F60H单元中

▶ 练习题：

- 1) 给定段地址0001H，通过改变偏移地址寻址，CPU可以寻址的范围是从__到__。
- 2) 有一数据存放在20000H中，现给定段地址SA，若想要用偏移地址寻到此单元数据。SA应满足什么条件：最小为__，最大为__。
- 3) 第2题中，当SA在什么范围内取值时，CPU无论如何都无法访问到20000H单元？

二. 存储器(组织与结构)

▶ 2. 实际地址的产生

- 实际地址的表示：

▶ 例：数据在21F60H中

- 1) 数据在2000:1F60内存单元中
- 2) 数据在2000H段中的1F60H单元中

▶ 练习题：

- 1) 给定段地址0001H，通过改变偏移地址寻址，CPU可以寻址的范围是从 00010H 到 1000F。
- 2) 有一数据存放在20000H中，现给定段地址SA，若想要用偏移地址寻到此单元数据。SA应满足什么条件：最小为 1001H，最大为 2000H。
- 3) 第2题中，当SA在什么范围内取值时，CPU无论如何都无法访问到20000H单元？

一. 微处理器的结构(8086/8088)

▶ 测试1：

- **标志寄存器* (F)**

➤ 计算下列指令执行后，OF、SF、ZF、AF、PF和CF各标志位的状态。

行号	程序指令	CF(after)	OF(after)	SF(after)	ZF(after)	PF(after)	AF(after)
1	SUB AL, AL						
2	MOV AL, 10H						
3	ADD AL, 90H						
4	MOV AL, 80H						
5	ADD AL, 80H						
6	MOV AL, FCH						
7	ADD AL, 05H						
8	MOV AL, 7DH						
9	ADD AL, 0BH						