東北大學
Northeastern University

汇编语言程序设计

主讲：东北大学计算机学院　刘松冉

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)

内容回顾

1. 通用寄存器
   - 使用
   - 特殊性
2. 标志寄存器
   - OF, ZF, CF, AF, PF, SF
3. 段寄存器
   - 为什么分段？
   - 存储什么内容
4. 存储器的组织与结构
   - 存储器分段技术
   - 实际地址生成

指令名称	格式	操作
MOV	MOV dest, src	dest ← src
ADD	ADD dest, src	dest ← dest+src
SUB	SUB dest, src	dest ← dest–src

指令名称	格式	操作
MUL	MUL src	AX ← AL * src (字节乘法) DX, AX ← AX * src (字乘

内容回顾

行号	程序指令	CF(after)	OF(after)	SF(after)	ZF(after)	PF(after)	AF(after)
1	**SUB AL, AL**	0(NC)	0(NV)	0(PL)	1(ZR)	1(PE)	0(NA)
2	**MOV AL, 10H**						
3	**ADD AL, 90H**	0(NC)	0(NV)	1(NG)	0(NZ)	1(PE)	0(NA)

FLAGS			SET (a 1-bit)	CLEARed (a 0-bit)
Overflow	of	=	OV (OVerflow)	NV [No oVerflow]
Direction	df	=	DN (decrement)	UP (increment)
Interrupt	if	=	EI (enabled)	DI (disabled)
Sign	sf	=	NG (negative)	PL (positive)
Zero	zf	=	ZR [zero]	NZ [Not zero]
Auxiliary Carry	af	=	AC	NA [No AC]
Parity	pf	=	PE (even)	PO (odd)
Carry	cf	=	CY [Carry]	NC [No Carry]

第三章 微型计算机的结构

三. 寄存器再学习

1. CS 和 IP
2. DS
3. SS 和 SP

三. 寄存器再学习

1. CS 和 IP

- CS和IP是8086CPU中最关键的寄存器，它们指示了CPU当前要读取指令的地址。
  - CS：代码段寄存器
  - IP：指令指针寄存器
- 8086PC工作过程的简要描述

1）从CS:IP指向内存单元读取指令，读取的指令进入指令缓冲器

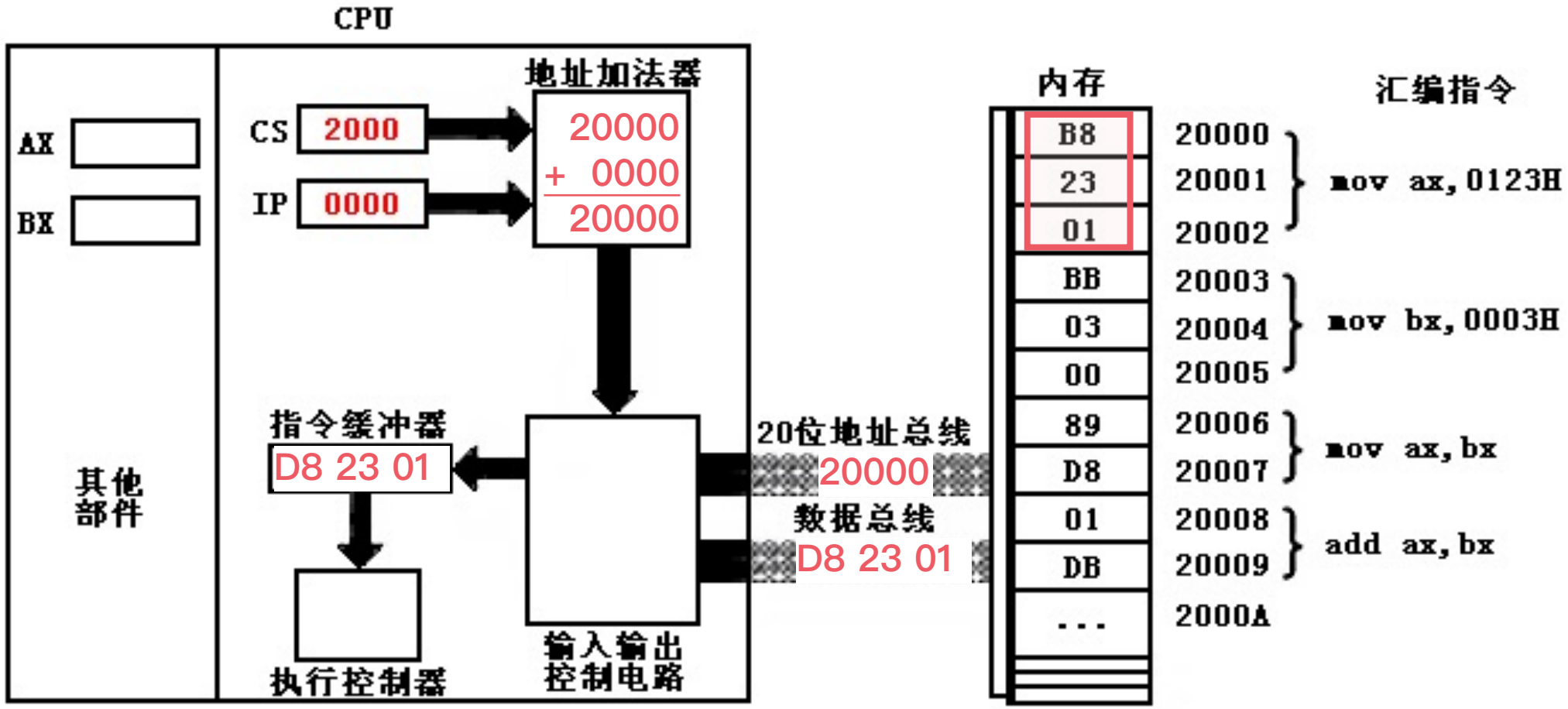
2）IP = IP + 所读取指令的长度，从而指向下一条指令

3）执行指令。 转到步骤1 ，重复这个过程

三. 寄存器再学习

1. CS 和 IP

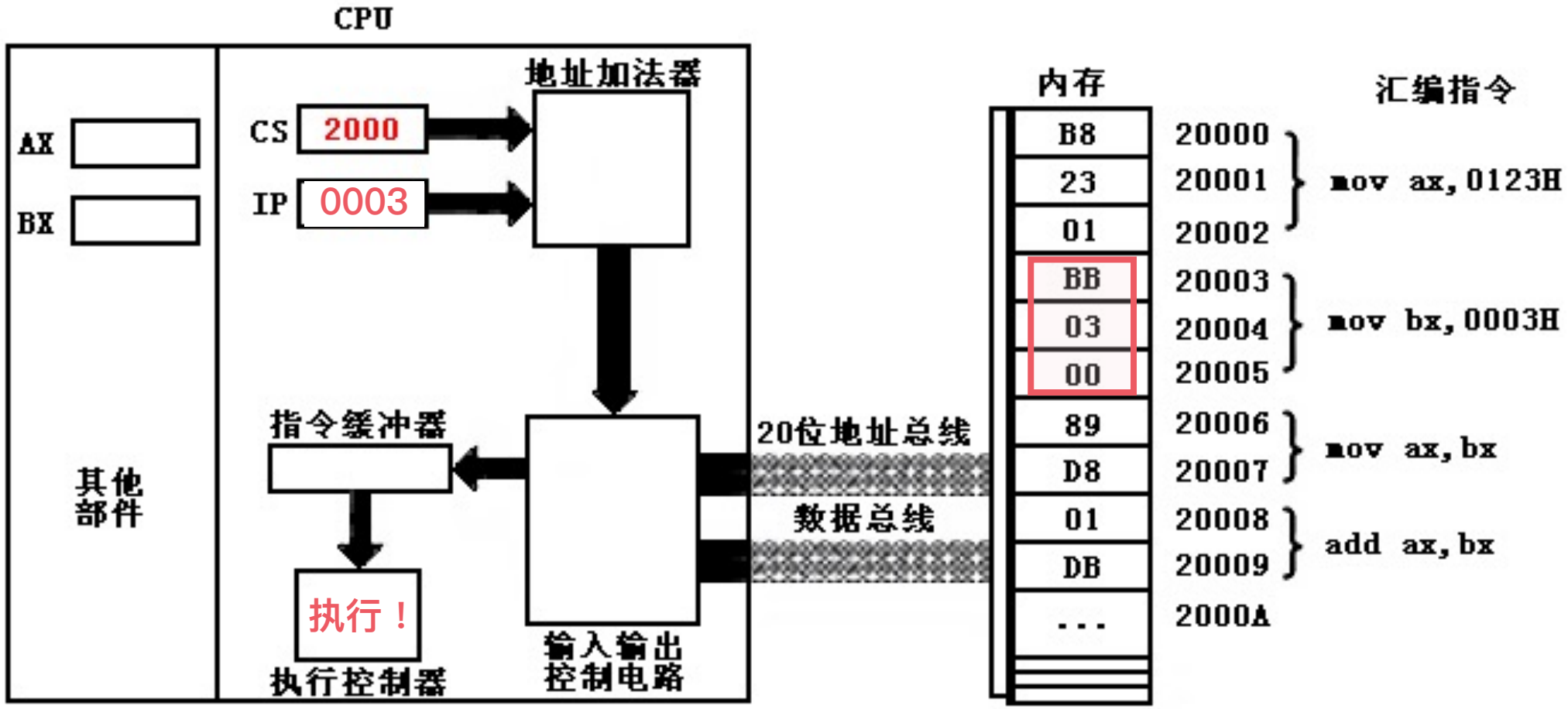
- 8086 CPU读取和执行指令演示



三. 寄存器再学习

1. CS 和 IP

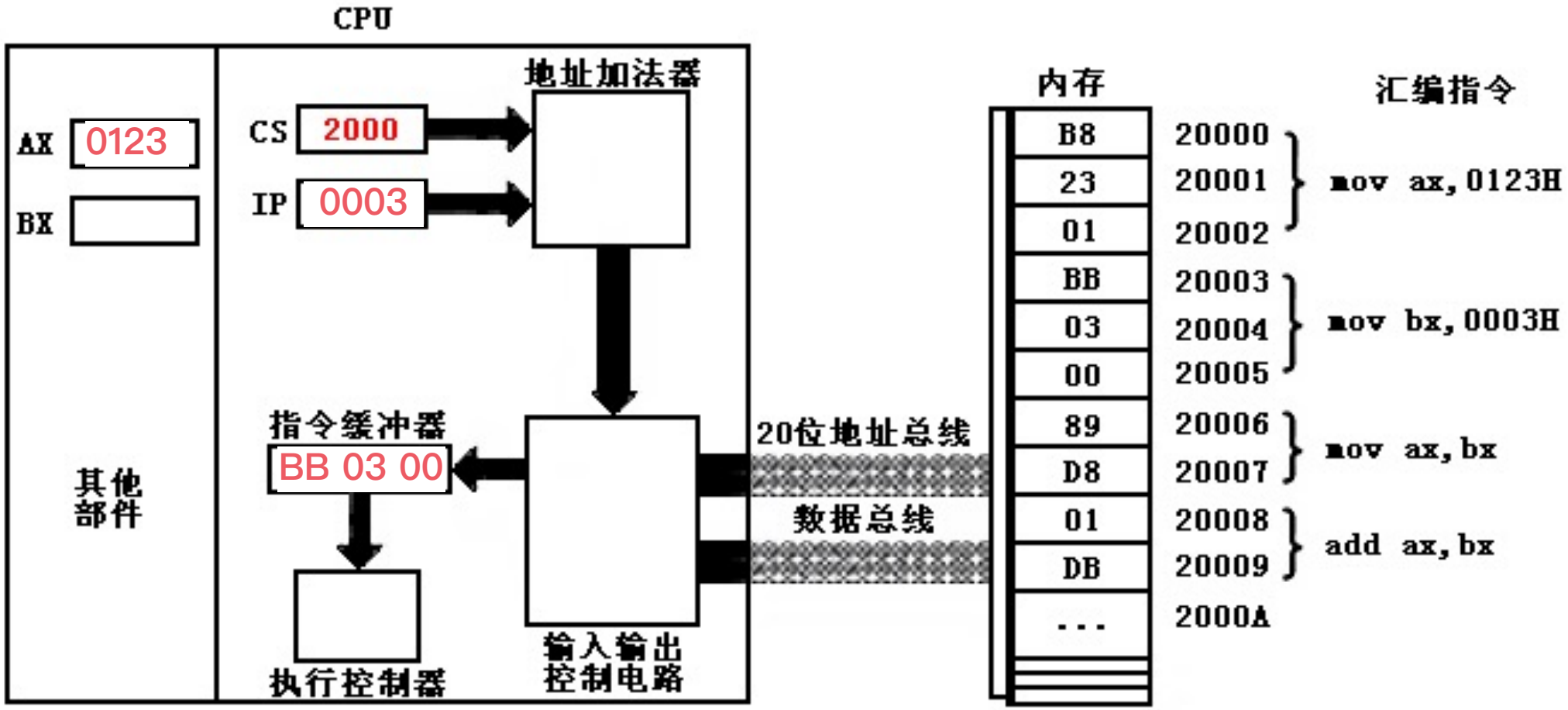
- 8086 CPU读取和执行指令演示



三. 寄存器再学习

1. CS 和 IP

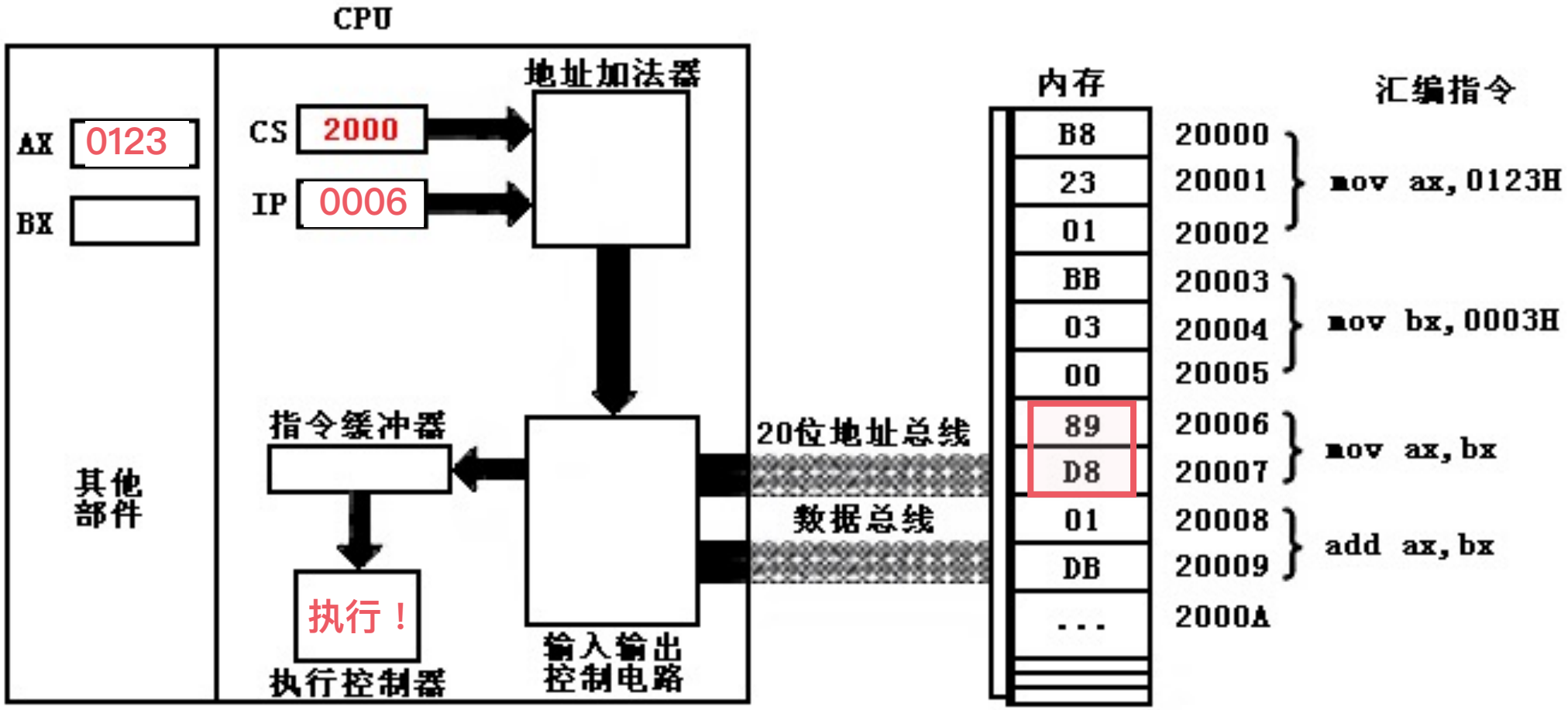
- 8086 CPU读取和执行指令演示



三. 寄存器再学习

1. CS 和 IP

- 8086 CPU读取和执行指令演示



1. CS 和 IP

- 补充信息：

1) 内存中指令和数据没有任何区别，都是二进制信息，CPU在工作的时候把有的信息看作指令，有的信息看作数据

2)CPU根据什么将内存中的信息看作指令？CPU将CS:IP指向的内存单元中的内容看作指令。

3)在任何时候，CPU将CS、IP中的内容当作指令的段地址和偏移地址，用它们合成指令的物理地址，到内存中读取指令码，执行。

4)如果说，内存中的一段信息曾被CPU执行过的话，那么，它所在的内存单元必然被CS:IP指向过

三. 寄存器再学习

1. CS 和 IP

- 如何修改CS、IP的值？
  - mov指令不能用于设置CS、IP的值，8086CPU没有提供这样的功能
  - 同时修改CS、IP的内容：JMP 段地址：偏移地址
    - JMP 2AE3:3
    - JMP 3:0B16
  - 仅修改IP的内容：JMP 某一合法寄存器
    - JMP AX
    - JMP BX

1. CS 和 IP

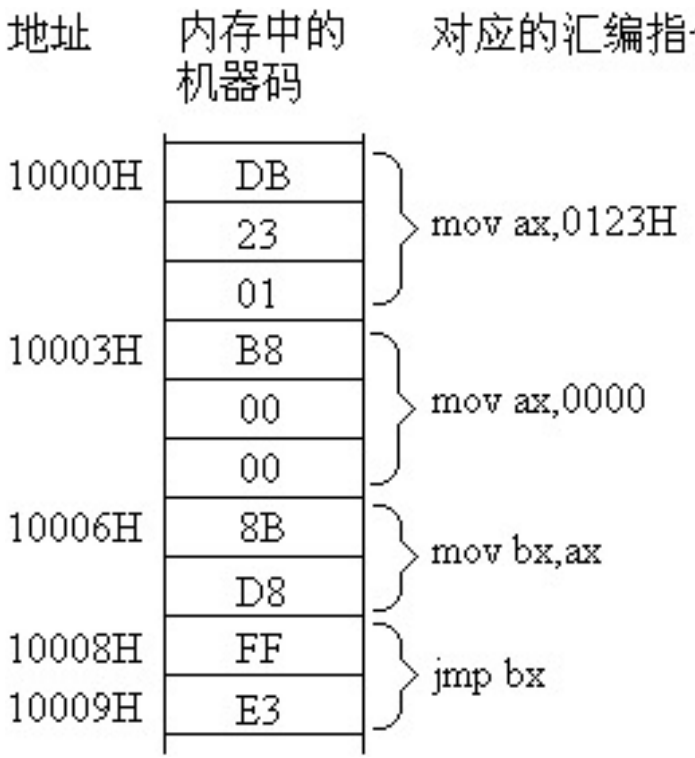
- 如何修改CS、IP的值？
  - 例子：请写出指令执行序列。（初始： CS=2000H，IP=0000H）

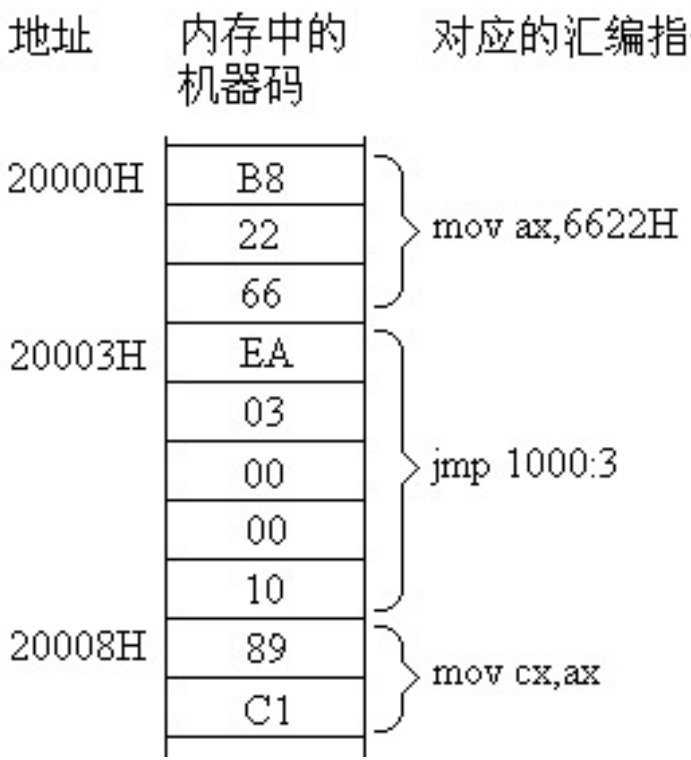
地址	内存中的机器码	对应的汇编指令
10000H	DB	mov ax,0123H
	23	
	01	
10003H	B8	mov ax,0000
	00	
	00	
10006H	8B	mov bx,ax
	D8	
10008H	FF	jmp bx
10009H	E3	

地址	内存中的机器码	对应的汇编指令
20000H	B8	mov ax,6622H
	22	
	66	
20003H	EA	jmp 1000:3
	03	
	00	
	00	
	10	
20008H	89	mov cx,ax
	C1	

1. CS 和 IP

- 如何修改CS、IP的值？
  - 例子：请写出指令执行序列。（初始： CS=2000H，IP=0000H）

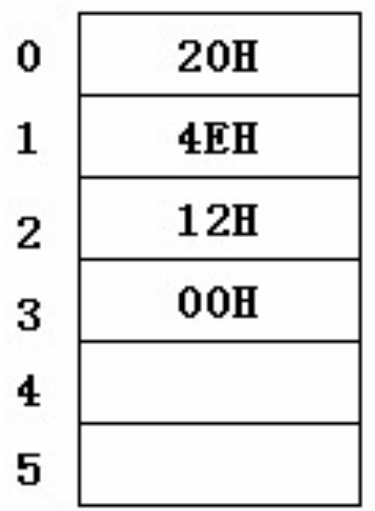




行号	程序指令
1	MOV AX, 6622
2	JMP 1000: 3
3	MOV AX, 0000
4	MOV BX, AX
5	JMP BX
6	MOV AX, 0123H
7	转到 第3行执行

2. DS：数据段寄存器

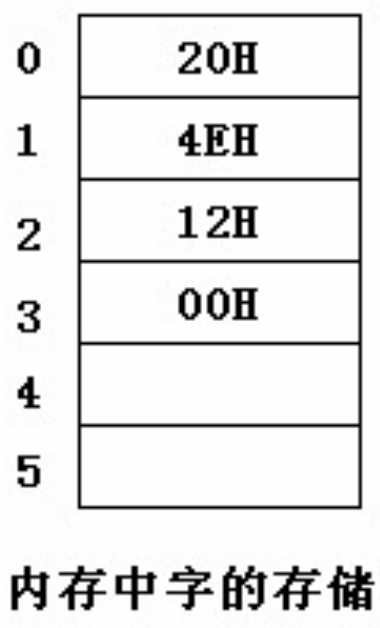
- 8086内存中字的存储（高8位存放在高位字节，低8位在低位字节）
  - 例子：在0地址处开始存放20000（4E20H）
  - 提问：
    1) 0地址单元中存放的字节型数据是多少？
    2) 0地址字单元中存放的字型数据是多少？
    3) 2地址字单元中存放的字节型数据是多少？
    4) 2地址单元中存放的字型数据是多少？
    5) 1地址字单元中存放的字型数据是多少？



内存中字的存储

2. DS：数据段寄存器

- 8086内存中字的存储（高8位存放在高位字节，低8位在低位字节）
  - 例子：在0地址处开始存放20000（4E20H）
  - 提问：
    1) 0地址单元中存放的字节型数据是多少？
    2) 0地址字单元中存放的字型数据是多少？
    3) 2地址字单元中存放的字节型数据是多少？
    4) 2地址单元中存放的字型数据是多少？
    5) 1地址字单元中存放的字型数据是多少？



内存中字的存储

  - 结论：任何两个地址连续的内存单元，N号单元和 N+1号单元，可以将它们看成两个独立的字节单元 ，也可以看成一个地址为N的字单元中的高位字节单元和低位字节单元。

2. DS：数据段寄存器

- 存储器（内存）中数据的读写
  - CPU通过内存地址读写数据
  - 在8086PC中，内存地址由段地址和偏移地址组成
  - DS寄存器存放要访问的数据的段地址

2. DS：数据段寄存器

- 存储器（内存）中数据的读写
  - CPU通过内存地址读写数据
  - 在8086PC中，内存地址由段地址和偏移地址组成
  - DS寄存器存放要访问的数据的段地址
  - 例1：读取10003H（1000:3）单元的内容

行号	程序指令
1	MOV BX, 1000H
2	MOV DS, BX
3	MOV AL, [3]

1) “[…]”表示一个内存单元，“[…]”中的…表示内存单元的偏移地址
2) 8086CPU不支持将数据直接送入DS段寄存器的操作，即mov DS,1000H是非法的。（硬件设计的问题）
3) DS赋值步骤：数据→一般寄存器→DS

2. DS：数据段寄存器

- 存储器（内存）中数据的读写
  - CPU通过内存地址读写数据
  - 在8086PC中，内存地址由段地址和偏移地址组成
  - DS寄存器存放要访问的数据的段地址
  - 例1：读取10003H（1000:3）单元的内容
  - 例2：将AL中的数据送入内存单元10000H

2. DS：数据段寄存器

- 存储器（内存）中数据的读写
  - CPU通过内存地址读写数据
  - 在8086PC中，内存地址由段地址和偏移地址组成
  - DS寄存器存放要访问的数据的段地址
  - 例1：读取10003H（1000:3）单元的内容
  - 例2：将AL中的数据送入内存单元10000H

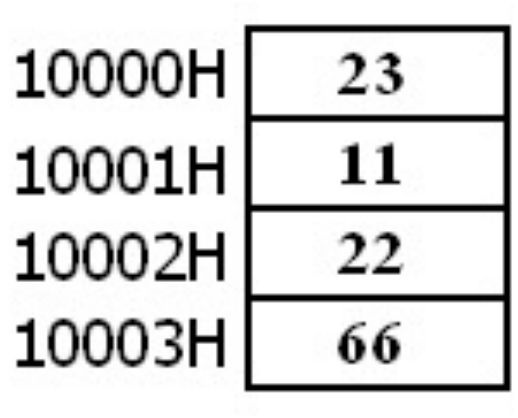
行号	程序指令
1	MOV BX, 1000H
2	MOV DS, BX
3	MOV [0], AL

一种合理的回答

2. DS：数据段寄存器

- 存储器（内存）中数据的读写
  - 例3：内存中的情况如右图，写出下面指令执行后寄存器AX，BX，CX中的值

行号	程序指令	
1	MOV	AX, 1000H
2	MOV	DS, AX
3	MOV	AX, [0]
4	MOV	BX, [2]
5	MOV	CX, [1]
6	ADD	BX, [1]
7	ADD	CX, [2]

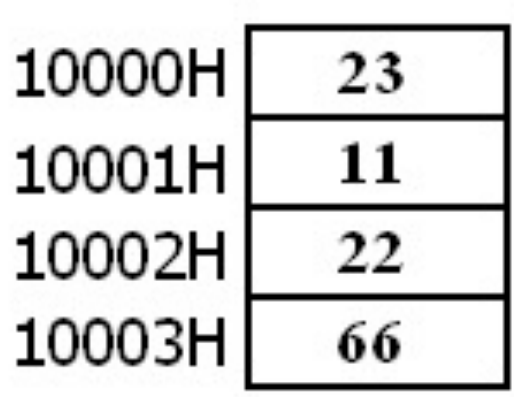


内存情况示意

2. DS：数据段寄存器

- 存储器（内存）中数据的读写
  - 例3：内存中的情况如右图，写出下面指令执行后寄存器AX，BX，CX中的值

行号	程序指令		After
1	MOV	AX, 1000H	AX = 1000H
2	MOV	DS, AX	DS = 1000H
3	MOV	AX, [0]	AX = 1123H
4	MOV	BX, [2]	BX = 6622H
5	MOV	CX, [1]	CX = 2211H
6	ADD	BX, [1]	BX = 8833H
7	ADD	CX, [2]	CX = 8833H



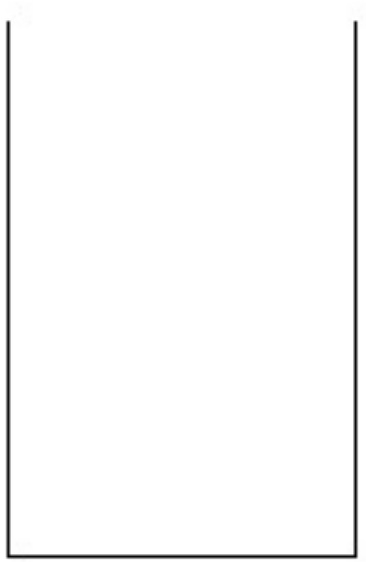
内存情况示意

3. SS：堆栈段寄存器

- 栈是一种具有特殊的访问方式的存储空间。它的特殊性就在于，最后进入这个空间的数据，最先出去。

3. SS：堆栈段寄存器

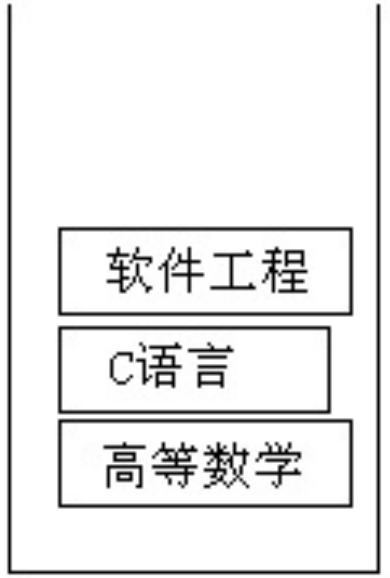
- 栈是一种具有特殊的访问方式的存储空间。它的特殊性就在于，最后进入这个空间的数据，最先出去。
- 入栈和出栈（Last In First Out, LIFO）
  - 入栈：将一个新的元素放到栈顶



入栈的方式

3. SS：堆栈段寄存器

- 栈是一种具有特殊的访问方式的存储空间。它的特殊性就在于，最后进入这个空间的数据，最先出去。
- 入栈和出栈（Last In First Out, LIFO）
  - 入栈：将一个新的元素放到栈顶
  - 出栈：从栈顶取出一个元素



出栈的方式

3. SS：堆栈段寄存器

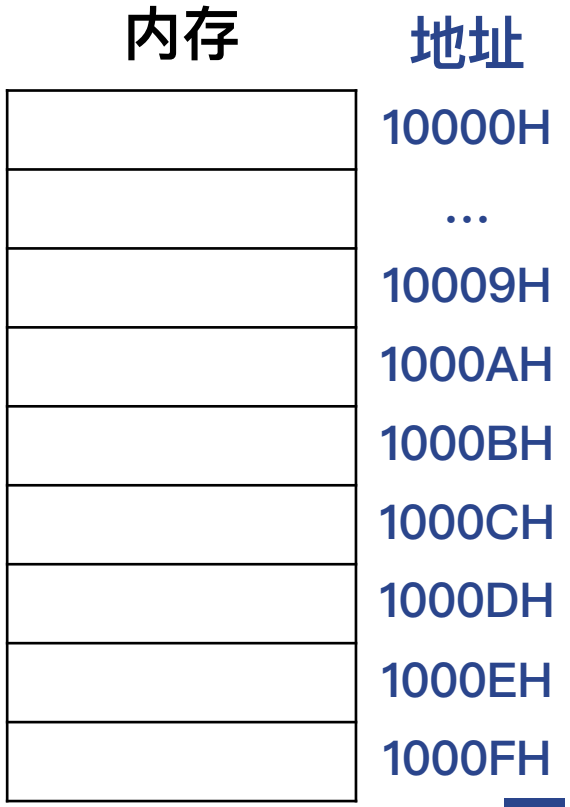
- **8086 CPU提供的栈机制**：8086CPU提供相关的指令来以栈的方式访问内存空间 → 在基于8086CPU编程的时候，可以将一段内存当作栈来使用。
- 8086CPU提供入栈和出栈指令
  - PUSH(入栈)，POP(出栈)
  - 入栈和出栈操作都是以字为单位进行的

指令名称	格式	操作	说明	指令分类
PUSH	PUSH src	(SP+1,SP) ← src SP ← SP-2	push word onto stack（进栈）	数据传送
POP	POP dest	dest ← (SP+1,SP) SP ← SP+2	Pop word off stack into destination（出栈）	算数运算

3. SS：堆栈段寄存器

- 8086CPU提供入栈和出栈指令
  - PUSH(入栈)，POP(出栈)
  - 入栈和出栈操作都是以字为单位进行的
- 例：（将10000H~1000FH）这段内存当做栈使用。

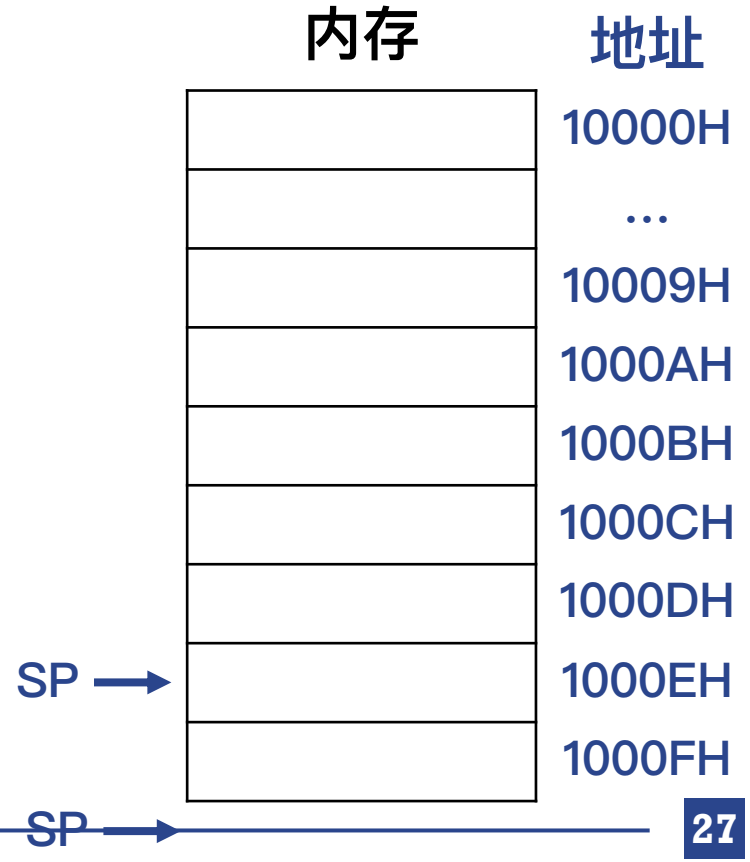
行号	程序指令		After
1	MOV	AX, 0123H	
2	PUSH	AX	
3	MOV	BX, 2266H	
4	PUSH	BX	
5	MOV	CX, 1122H	
6	PUSH	CX	
7	POP	AX	
8	POP	BX	
9	POP	CX	



3. SS：堆栈段寄存器

- 8086CPU提供入栈和出栈指令
  - PUSH(入栈)，POP(出栈)
  - 入栈和出栈操作都是以字为单位进行的
- 例：（将10000H~1000FH）这段内存当做栈使用。

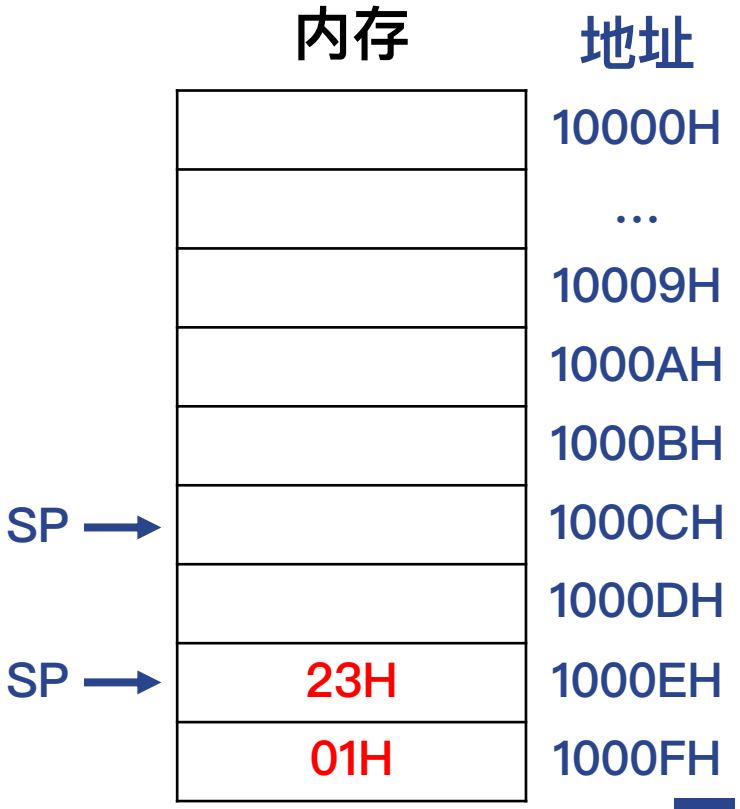
行号	程序指令		After
1	MOV	AX, 0123H	AX=0123H
2	PUSH	AX	
3	MOV	BX, 2266H	
4	PUSH	BX	
5	MOV	CX, 1122H	
6	PUSH	CX	
7	POP	AX	
8	POP	BX	
9	POP	CX	



3. SS：堆栈段寄存器

- 8086CPU提供入栈和出栈指令
  - PUSH(入栈)，POP(出栈)
  - 入栈和出栈操作都是以字为单位进行的
- 例：（将10000H~1000FH）这段内存当做栈使用。

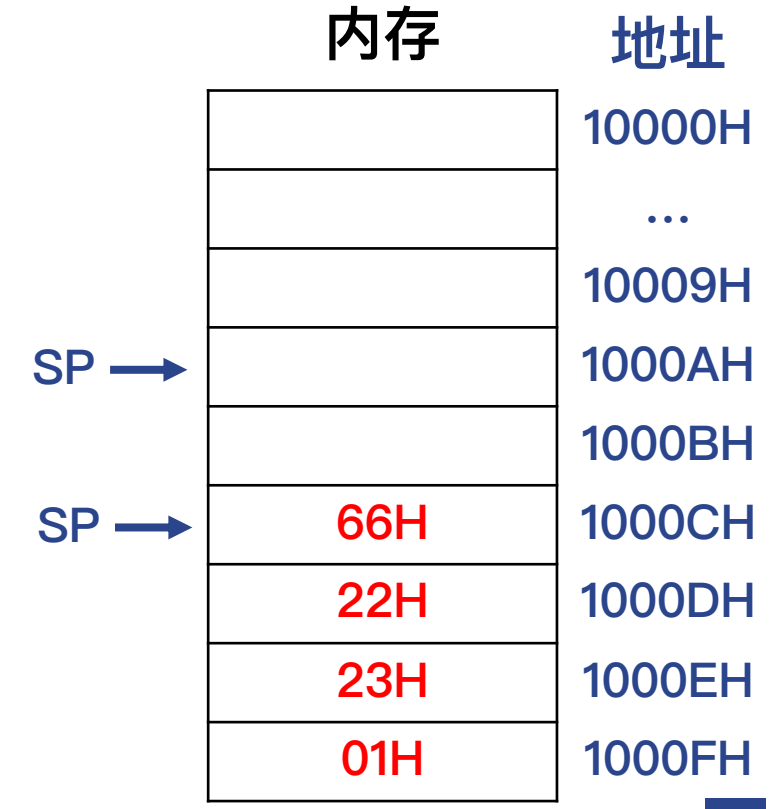
行号	程序指令		After
1	MOV	AX, 0123H	AX=0123H
2	PUSH	AX	
3	MOV	BX, 2266H	
4	PUSH	BX	
5	MOV	CX, 1122H	
6	PUSH	CX	
7	POP	AX	
8	POP	BX	
9	POP	CX	



3. SS：堆栈段寄存器

- 8086CPU提供入栈和出栈指令
  - PUSH(入栈)，POP(出栈)
  - 入栈和出栈操作都是以字为单位进行的
- 例：（将10000H~1000FH）这段内存当做栈使用。

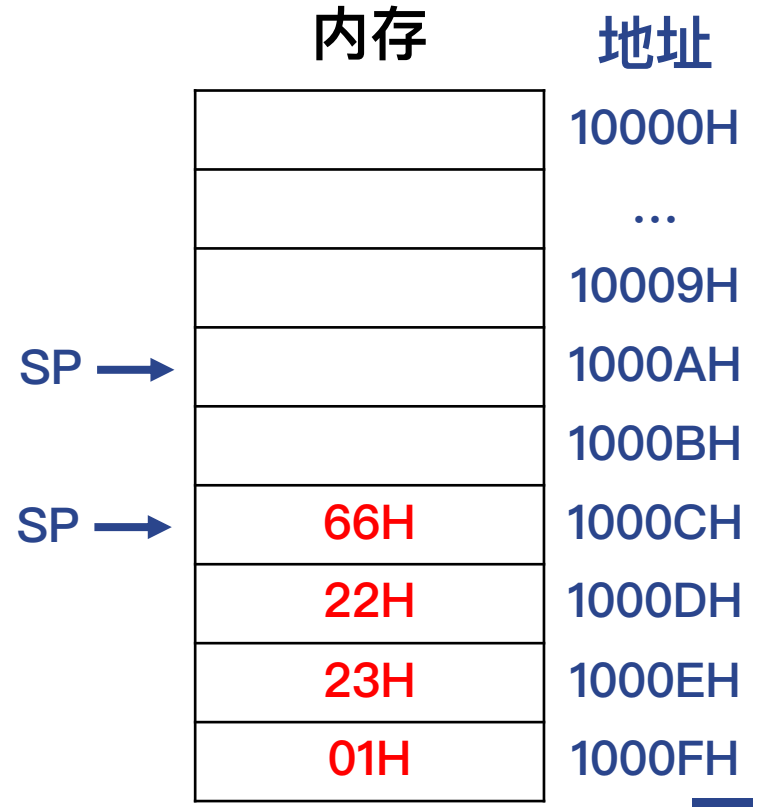
行号	程序指令		After
1	MOV	AX, 0123H	AX=0123H
2	PUSH	AX	
3	MOV	BX, 2266H	BX=2266H
4	PUSH	BX	
5	MOV	CX, 1122H	
6	PUSH	CX	
7	POP	AX	
8	POP	BX	
9	POP	CX	



3. SS：堆栈段寄存器

- 8086CPU提供入栈和出栈指令
  - PUSH(入栈)，POP(出栈)
  - 入栈和出栈操作都是以字为单位进行的
- 例：（将10000H~1000FH）这段内存当做栈使用。

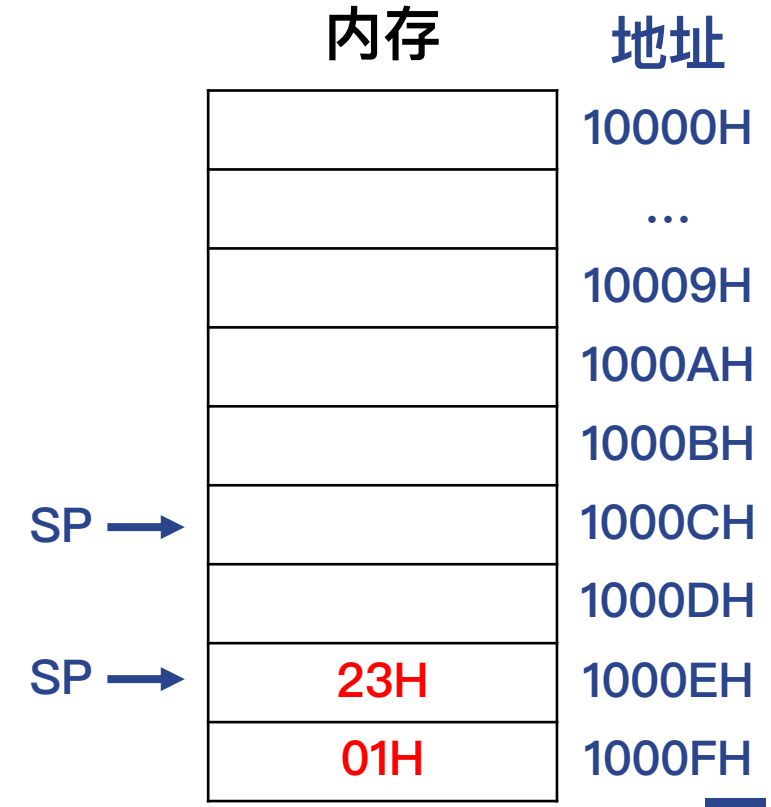
行号	程序指令		After
1	MOV	AX, 0123H	AX=0123H
2	PUSH	AX	
3	MOV	BX, 2266H	BX=2266H
4	PUSH	BX	
5	MOV	CX, 1122H	CX=1122H
6	PUSH	CX	
7	POP	AX	
8	POP	BX	
9	POP	CX	



3. SS：堆栈段寄存器

- 8086CPU提供入栈和出栈指令
  - PUSH(入栈)，POP(出栈)
  - 入栈和出栈操作都是以字为单位进行的
- 例：（将10000H~1000FH）这段内存当做栈使用。

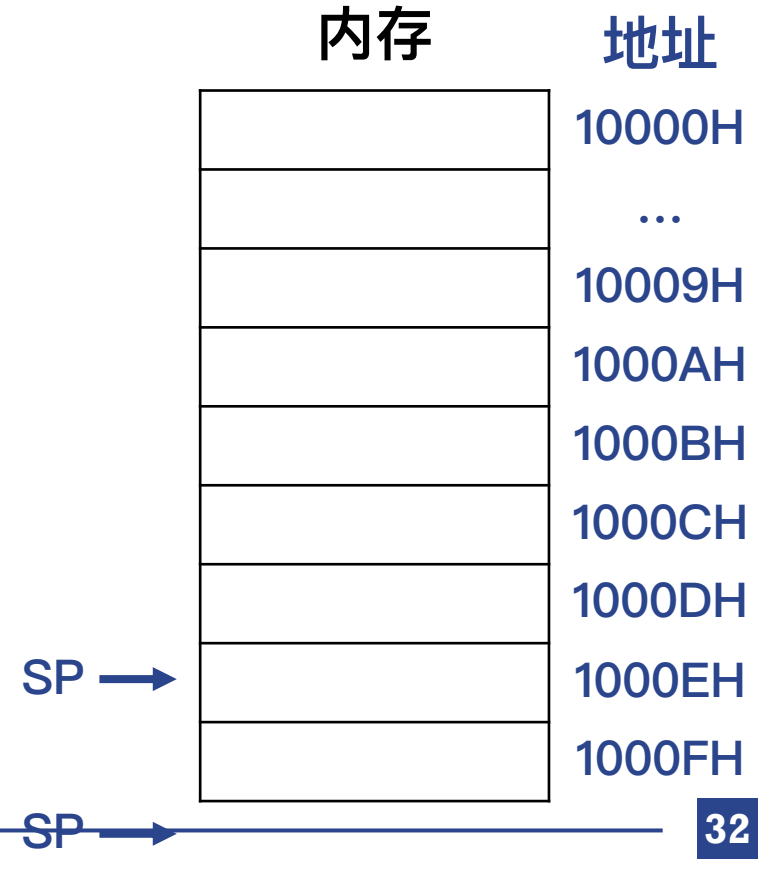
行号	程序指令		After
1	MOV	AX, 0123H	AX=0123H
2	PUSH	AX	
3	MOV	BX, 2266H	BX=2266H
4	PUSH	BX	
5	MOV	CX, 1122H	CX=1122H
6	PUSH	CX	
7	POP	AX	AX=1122H
8	POP	BX	
9	POP	CX	



3. SS：堆栈段寄存器

- 8086CPU提供入栈和出栈指令
  - PUSH(入栈)，POP(出栈)
  - 入栈和出栈操作都是以字为单位进行的
- 例：（将10000H~1000FH）这段内存当做栈使用。

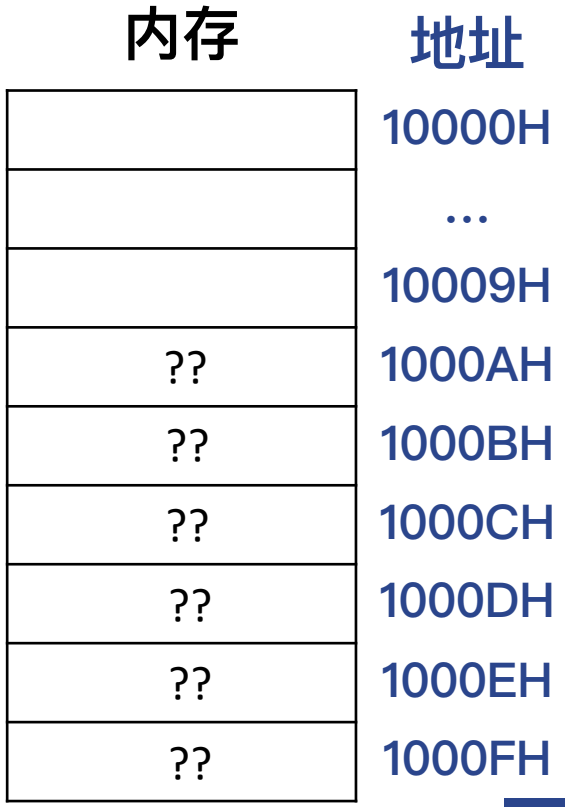
行号	程序指令		After
1	MOV	AX, 0123H	AX=0123H
2	PUSH	AX	
3	MOV	BX, 2266H	BX=2266H
4	PUSH	BX	
5	MOV	CX, 1122H	CX=1122H
6	PUSH	CX	
7	POP	AX	AX=1122H
8	POP	BX	BX=2266H
9	POP	CX	



3. SS：堆栈段寄存器

- 8086CPU提供入栈和出栈指令
  - PUSH(入栈)，POP(出栈)
  - 入栈和出栈操作都是以字为单位进行的
- 例：（将10000H~1000FH）这段内存当做栈使用。

行号	程序指令		After
1	MOV	AX, 0123H	AX=0123H
2	PUSH	AX	
3	MOV	BX, 2266H	BX=2266H
4	PUSH	BX	
5	MOV	CX, 1122H	CX=1122H
6	PUSH	CX	
7	POP	AX	AX=1122H
8	POP	BX	BX=2266H
9	POP	CX	CX=0123H



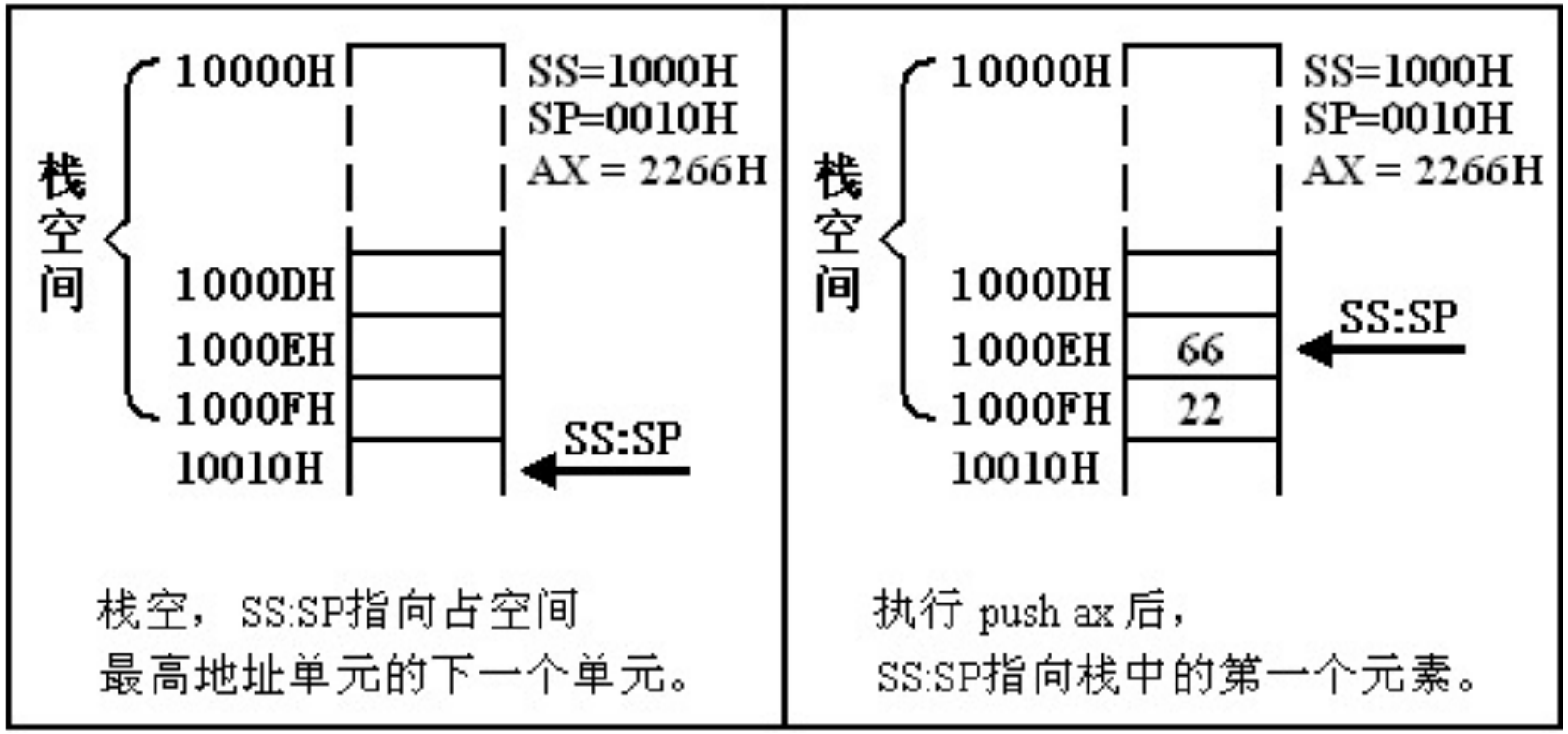
3. SS：堆栈段寄存器

- **疑问**：
  - CPU如何知道一段内存空间被当作栈使用？
    - SS指示
  - 执行push和pop的时候，如何知道哪个单元是栈顶单元？
    - 任意时刻，SS:SP指向栈顶元素
    - 段寄存器SS，存放栈顶的段地址
    - 寄存器SP，存放栈顶的偏移地

3. SS：堆栈段寄存器

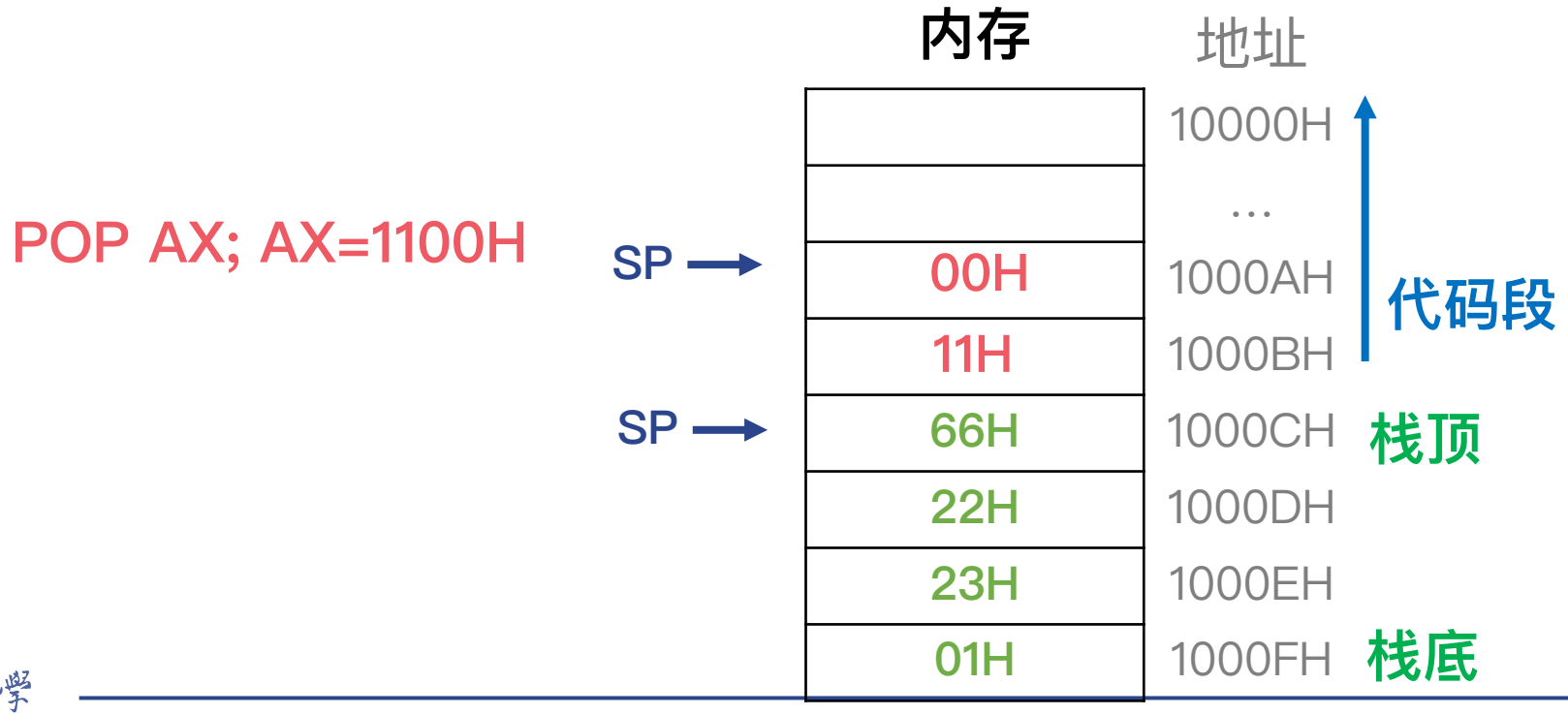
- **练习1：**
  - 如果我们将10000H ~ 1000FH 这段空间当作栈，初始状态栈是空的，此时，SS=1000H，SP= ？

SP = 0010H



3. SS：堆栈段寄存器

- SS和SP只记录了栈顶的地址，依靠SS和SP可以保证在入栈和出栈时找到栈顶。
- 如何能够保证在入栈、出栈时，栈顶不会超出栈空间？
  - 当栈满的时候再使用push指令入栈，栈空的时候再使用pop指令出栈，都将发生栈顶超界问题。栈顶超界是危险的。



3. SS：堆栈段寄存器

- SS和SP只记录了栈顶的地址，依靠SS和SP可以保证在入栈和出栈时找到栈顶。
- 如何能够保证在入栈、出栈时，栈顶不会超出栈空间？
  - 当栈满的时候再使用push指令入栈，栈空的时候再使用pop指令出栈，都将发生栈顶超界问题。栈顶超界是危险的。
  - 因为我们既然将一段空间安排为栈 ，那么在栈空间之外的空间里很可能存放了具有其他用途的数据、代码等，这些数据、代码可能是我们自己的程序中的，也可能是别的程序中的。（毕竟一个计算机系统并不是只有我们自己的程序在运行）

3. SS：堆栈段寄存器

- SS和SP只记录了栈顶的地址，依靠SS和SP可以保证在入栈和出栈时找到栈顶。
- 如何能够保证在入栈、出栈时，栈顶不会超出栈空间？
  - 当栈满的时候再使用push指令入栈，栈空的时候再使用pop指令出栈，都将发生栈顶超界问题。栈顶超界是危险的。
  - 因为我们既然将一段空间安排为栈 ，那么在栈空间之外的空间里很可能存放了具有其他用途的数据、代码等，这些数据、代码可能是我们自己的程序中的，也可能是别的程序中的。（毕竟一个计算机系统并不是只有我们自己的程序在运行）
  - **结论**：我们在编程的时候要自己操心栈顶超界的问题 ，要根据可能用到的最大栈空间，来安排栈的大小，防止入栈的数据太多而导致的超界；执行出栈操作的时候也要注意，以防栈空的时候继续出栈而导致的超界。

3. SS：堆栈段寄存器

- **练习2**：
  - ➢编程：
    1) 将10000H~1000FH 这段空间当作栈，初始状态是空的；
    2) 设置AX=001AH，BX=001BH；
    3) 将AX、BX中的数据入栈；
    4) 然后将AX、BX清零；
    5) 从栈中恢复AX、BX原来的内容。

行号	程序指令	
1	MOV	AX, 1000H
2	MOV	SS, AX
3	MOV	SP, 0010H
4	MOV	AX, 001AH
5	MOV	BX, 001BH
6	PUSH	AX
7	PUSH	BX
8	SUB	AX, AX
9	SUB	BX, BX
10	POP	BX
11	POP	AX

3. SS：堆栈段寄存器

- **练习3**：
  - ➢编程：
    1) 将10000H~1000FH 这段空间当作栈，初始状态是空的；
    2) 设置AX=002AH，BX=002BH；
    3) 利用栈 ，交换 AX 和 BX 中的数据。

行号	程序指令	
1	MOV	AX, 1000H
2	MOV	SS, AX
3	MOV	SP, 0010H
4	MOV	AX, 002AH
5	MOV	BX, 002BH
6	PUSH	AX
7	PUSH	BX
8	POP	AX
9	POP	BX

第三章 微型计算机的结构

四. 寻址方式

1. 操作数的种类
2. 寻址方式
3. 段更换与段跨越
4. 有效地址的计算时间
5. 指令系统

寻址方式：指令中提供操作数或操作数地址的方法。寻址方式是规定如何对指令代码中操作数字段进行解释以找到操作数的方法。

四. 寻址方式

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 （指令代码B010）

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]（指令代码A03412）

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（指令代码88D8）

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（指令代码E420）

2. 寻址方式 – 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。

2. 寻址方式 – 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 （其指令代码B83412）

2. 寻址方式 – 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器中
- 例子：

```
INC   CX        ; 指令码 41
INC   DX        ; 指令码 42
INC   BX        ; 指令码 43
INC   SP        ; 指令码 44
INC   BP        ; 指令码 45
MOV CX, BX      ; 指令码 89D9H
```

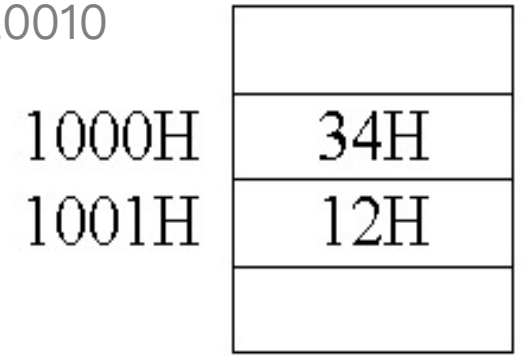
2. 寻址方式 – 存储器寻址

- 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

2. 寻址方式 – 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] ；指令码：A0010
MOV  BX, [1000H] ；指令码：8B1E0010
```



数据段寄存器DS=0000H

2. 寻址方式 – 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]  ；指令码8A07，偏移地址在BX中
MOV  BL, [SI]  ；指令码8B04
```

1000H	34H
1001H	12H

Case1: BX=1001H, SI = 1000H

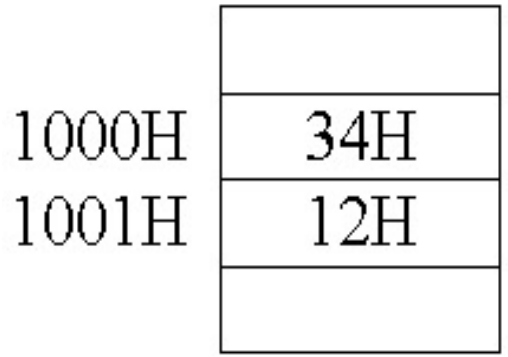
Case2: BX=1000H, SI = 1000H

2. 寻址方式 – 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] ；指令码8A07

MOV BL, [SI] ；指令码8B04



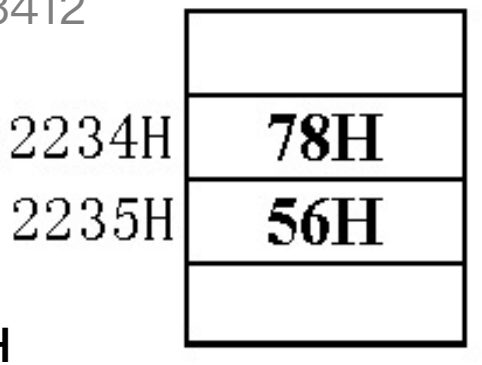
Case1: BX=1001H, SI = 1000H

Case2: BX=1002H, SI = 1000H

2. 寻址方式 – 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] ;指令码:8A873412



BX=1000H

数据段寄存器DS=0000H

2. 寻址方式 – 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

  例：用Debug查看内存，结果如下：

  2000:1000 BE 00 06 00 00 00 ……

  写出下面的程序执行后，ax、bx、cx中的内容。

```
mov ax,2000H
mov ds,ax
mov bx,1000H
mov ax,[bx]
mov cx,[bx+1]
add cx,[bx+2]
```

(ax)=00BEH
(bx)=1000H
(cx)=0606H

2. 寻址方式 – 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 ）

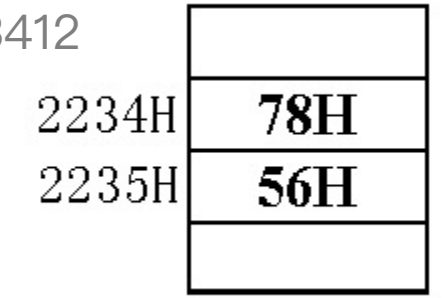
MOV AL, [BX+100H]
MOV AL, 100H[BX]

2. 寻址方式 – 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] ;指令码:8A843412

SI = 1000H



2. 寻址方式 – 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]

2. 寻址方式 – 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]；机器码8A803412

BX = 1000H，SI = 2000H

2. 寻址方式 – 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]

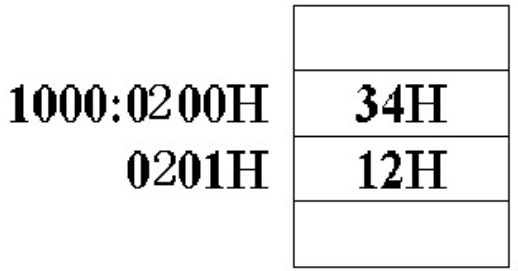
  位移量范围为补码表示的16位（ –32768~+32767 ）

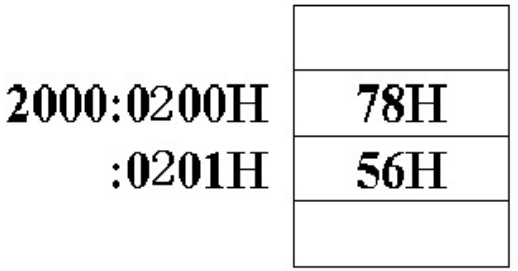
2. 寻址方式 – 存储器寻址

- 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
  - ➢ 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]





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**：寻址存储器操作数时，如果不加段跨越前缀，则在规定的隐含段内进行寻址，如果加了段跨越前缀，即可对段跨越前缀指出的段内进行寻址。例如：

```
MOV   AX, [BP]        ; AX ← SS:[BP,BP+1]
MOV   AX, DS:[BP]     ; AX ← DS:[BP,BP+1]
MOV   BL, [SI]        ; BL ← DS:[SI,SI+1]
MOV   BL, CS:[SI]     ; BL ← CS:[SI,SI+1]
```

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