Registers

Registers

The processor 8086 has four sets of 16-bit registers that the programmer can access it:

- IP Instruction Pointer register
- Four Data registers AX, BX, CX, DX.
- Four pointers and indexes registers SI, DI, BP, SP.
- Four segments registers CS, DS, SS, ES.
- In addition, another register is the Flag register and also called the status register. It is a 16-bit register, but only 9 digits are used.

Registers

The data inside the processor is stored in the registers.

The registers are divided into:

Data Registers : Data is handled in terms of storage, computational and logical operations.

Address Registers : where different addresses are stored.

Status Registers: It contains the status of the handler after executing a specific command.

The processor contains the number of 14 registers and we will in the next section explain the names and the function of each registerer.

Data Registers

- These registers are used to cache the interim results during the execution of a program.
- Storing data in these registers enables us to access those data faster than if they were in memory.
- Data registers are divided into
  - Accumulator register is denoted by the symbol A.
  - Base registers are denoted by the symbol B.
  - Count registers are denoted by C.
  - Data registers are denoted by D.

- Each of the previous registers can be used as a 16-bit word. This is evidenced by typing the letter X after the name of the register or can be used as two 8-bit bytes. This is indicated by the letters H, L where:
- L for the lower address bytes, e.g. AL.
- H for the larger address bytes, e.g. BH.
- So both of these registers can be used for mathematical or logical instruction in the assembly language such as And, Add.
- For some instructions, such as programs that contain string instructions, they use certain registers, such as the use of register C, to store the number that represents the number of bytes on which string instruction will be executed (number of times the string instruction is repeated)

Data Registers DX, CX, BX, AX

- These four registers are used to manipulate data within the processor and the programmer can deal directly with these registers.
- Although the processor can handle data in memory, dealing with registers is much faster than handling memory (fewer pulses are needed), so we always prefer to deal with registers for their speed. This is why the number of registers in modern processors has increased.

- Each of these registers can be used as a single 16-bits unit or two units, each with a capacity of 8-bits, one of which is High and Low.
- For example, the AX register can be treated as a 16-bits register or handle the upper half (HIGH) AH as 8-bits register and low AL (LOW) register as 8-bits register and similarly with register D, C, B and thus we have eight 8-bits registers or four 16-bits registers.

Although the four registers are GENERAL PURPOSE REGISTERS so that they can be used in any general use, each register has its own use which is dealt with in the following section:

1-Accumulator AX register

AX is the preferred register for use in calculations, logic, data transfer, memory handling, and I / O ports.

Its used in generating programs that shorten and increasing program efficiency. For example, in a two-digit operation, one of the two numbers should be placed with the value to be output to a specified exit port, and then the value entered from the exit port specified in it will be read.

Generally, the AX register is treated as the most important register in the processor.

2 -Base Register BX

The BX register uses as memory, where some operations require memory to be handled with a specific pointer and its cursor values, are changed to perform a scan of a specified portion of memory as we shall see later.

3 -Count Register CX

- The CX register is used as a counter to control the frequency of a specified set of instructions. It is also used to repeat the register rotation process for a specified number of times.

4 -Data Register DX

- It is used for multiplication and division operations
- It is also used as an indicator of input and output ports when using input and output operations.

مسجلات المقاطع

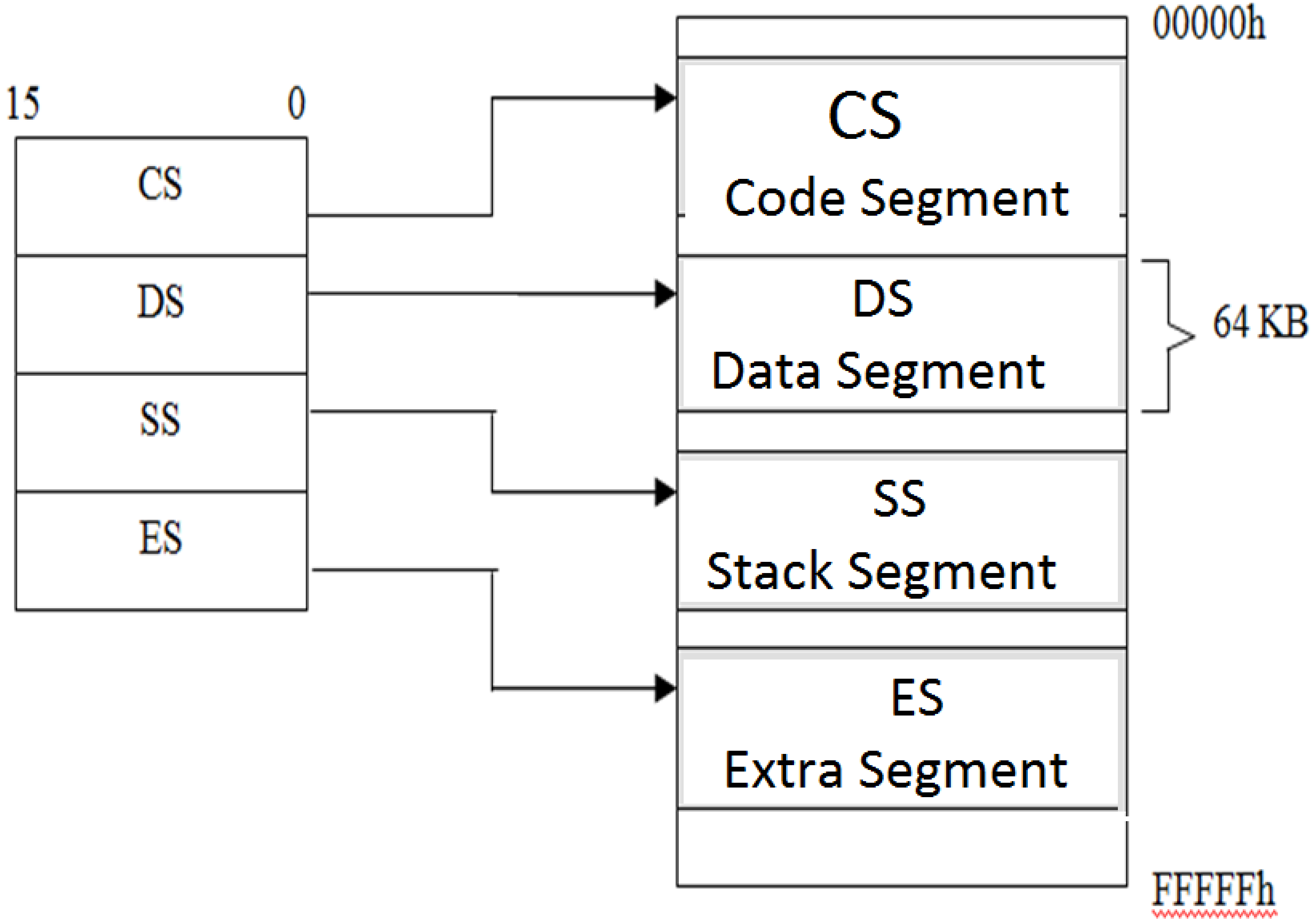
و هي عبارة عن أربعة مسجلات طول كل منها ١٦ بت أي ٢ بايت و هي :

١) مسجل مقطع الشيفرة CS : يحتوي على عنوان أول حجرة في مقطع شيفرة البرنامج في الذاكرة، أي أنه يشير إلى بداية مقطع الشيفرة.

٢) مسجل مقطع المعطيات DS : يحتوي على عنوان أول حجرة في مقطع المعطيات في الذاكرة، أي أنه يشير إلى بداية مقطع المعطيات.

٣) مسجل مقطع المكدس SS : يحتوي على عنوان أول حجرة في مقطع المكدس في الذاكرة، أي أنه يشير إلى بداية مقطع المكدس.

٤) مسجل مقطع المعطيات الإضافي ES : يحتوي على عنوان أول حجرة في مقطع المعطيات الإضافي في الذاكرة، أي أنه يشير إلى بداية مقطع المعطيات الإضافي.


15
0
CS
DS
SS
ES
00000h
CS
Code Segment
DS
Data Segment
64 KB
SS
Stack Segment
ES
Extra Segment
FFFFFh

CS, DS, SS, ES Registers

- These registers are used to specify a specific address in memory. In order to clarify the function of these registers, the method of organizing the memory must first be clarified. We know that the 8088 processor handles 20 address signals (the Address Bus has 20 signals) and thus can address memory up to 2^{20} = 1,048,576 ie 1 Mbytes.

00000 h	=	0000 0000 0000 0000 0000
00001 h	=	0000 0000 0000 0000 0001
00002 h	=	0000 0000 0000 0000 0010
00003 h	=	0000 0000 0000 0000 0011
00004 h	=	0000 0000 0000 0000 0100

- Because the titles in the binary form are very long, it is easier to deal with the addresses in the hexadecimal form, so the first address in the memory is 00000h and another address is FFFFFh.

Memory Segments

- The memory segment is a connected portion of 64 Kbytes.
- Each segment in memory is specified by a specific number called Segment Number, which starts with 0000h and ends with FFFFh.
- Inside the segment, the address is determined by a specific Offset and this offset is located after the specified location from the beginning of the segment, which is a number that is 16 bytes long and has a value between 0000h and FFFFh.

Code Segment Register CS

This register contains the title of the segment of the Code Segment Address where a specific segment of memory is specified in which the program is placed, and then the address of the handler must be defined where the program will be executed; the title of this segment must be specified and placed in a special register called the Code Segment Register CS and the offset value is determined by using the instruction pointer register which will be discussed later.

Data segment Register DS

This register contains the Data Segment Address where the data of the program is defined in a specific area of memory (the data segment is named) and the title of this segment is specified and placed in the DS register. You can then address the memory and deal with the different variables using other registers with the required displacement value.

Stack Segment Register SS

- A portion of the memory is identified and treated as a stack.
- Stack works in the Last In First Out LIFO method
- It is used in a number of processes, the most important of which is the call for subprogrammes as we shall see later.
- There are a set of registers are used to calculate the offset value, the most important of which is Stack Pointer SP.

Extra Segment Register ES

- This register is used to identify and address an additional segment. Sometimes you need to process more than one segment at a time (such as transferring a certain amount of data in memory from a specific location to another location in a remote segment, so the data register is not enough, but we need an extra register to determine the other segment.

Index and Pointer Registers (SP, BP, SI, DI)

These registers are used with the segment registers that we talked about it in the previous section of the communication with specific addresses in memory, and reverse segments registers can perform calculations and logic on these registers .

There are four auxiliary registers that help to find the physical address in cooperation with the registers, and the length of these registers are 16 bits or 2 bytes, namely:

1. Stack Pointer (SP)

This register is used with the stack segment and will talk in detail about the stack in the coming chapters.

2. Base Pointer (BP)

This register is used primarily to communicate with the data in the stack, but it reverses the stack pointer where it can be used to address memory in other segments than the stack segment.

3. Source Index (SI)

This register is used to address memory in the data segment where it points to the beginning (or end) of a specific area of memory that is required to be dealt with it. the value of this register is changed each time the entire area of memory is dealt with.

4. Destination Index (DI)

This register is used as the SI indexing register. This register points to the memory address in which the data will be stored. This is usually done with the ES extension segment register and there is a set of text commands that assume that the source address and destination address are specified in these registers.

Instruction Pointer (IP)

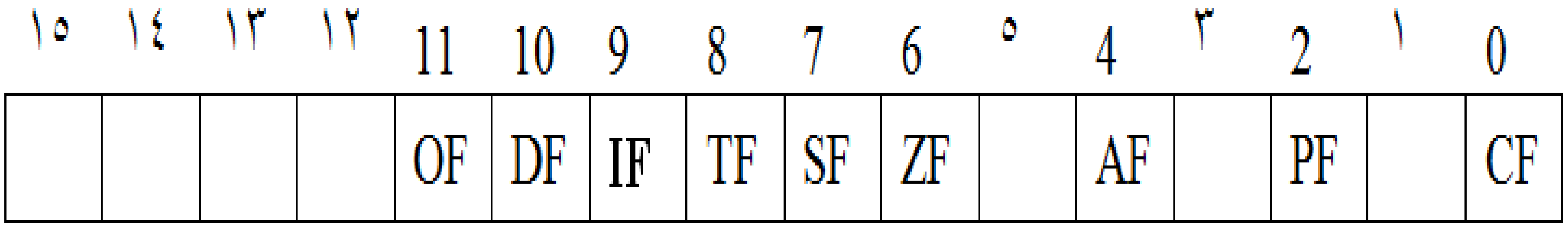
- All of the registers we talked about it so far are used to address data stored in memory.
- To address the program, we must know the address of the first command in the program to be executed. after that select, the address of the next command and continues to execute the program.
- The offset for the command to be executed is stored in the Instruction Pointer (IP), where this is done in the Code Segment, so the title of the command to be executed is CS: IP. The instruction indicator cannot be directly addressed from inside the program, but its value is changed indirectly, such as branching to a specific address where the value of that address is placed in the instruction index in case of branching.

Flags Register

- This register contains a set of flags which are two types: status and control flags.
- **The status flags**, it shows the status of the processor after performing each process to clarify the resulting state, where these flags can tell the result (e.g. if the zero is raised, meaning that the result of the last process equals zero).
- **The control flags** are used to notify the processor by doing specific something. For example, the Interrupt Flag can be used and the value is set to 0, so we ask the processor to ignore the calls

Flags Register

It is a 16-bit register that is present in the implementation unit as shown in Figure:



١٥	١٤	١٣	١٢	11	10	9	8	7	6	٥	4	٣	2	١	0
				OF	DF	IF	TF	SF	ZF		AF		PF		CF

As shown in the previous figure there are six flags of the status: CF, PF, AF, ZF, SF, OF, and there are three flags for the control DF, IF, TF.

A) Status flags

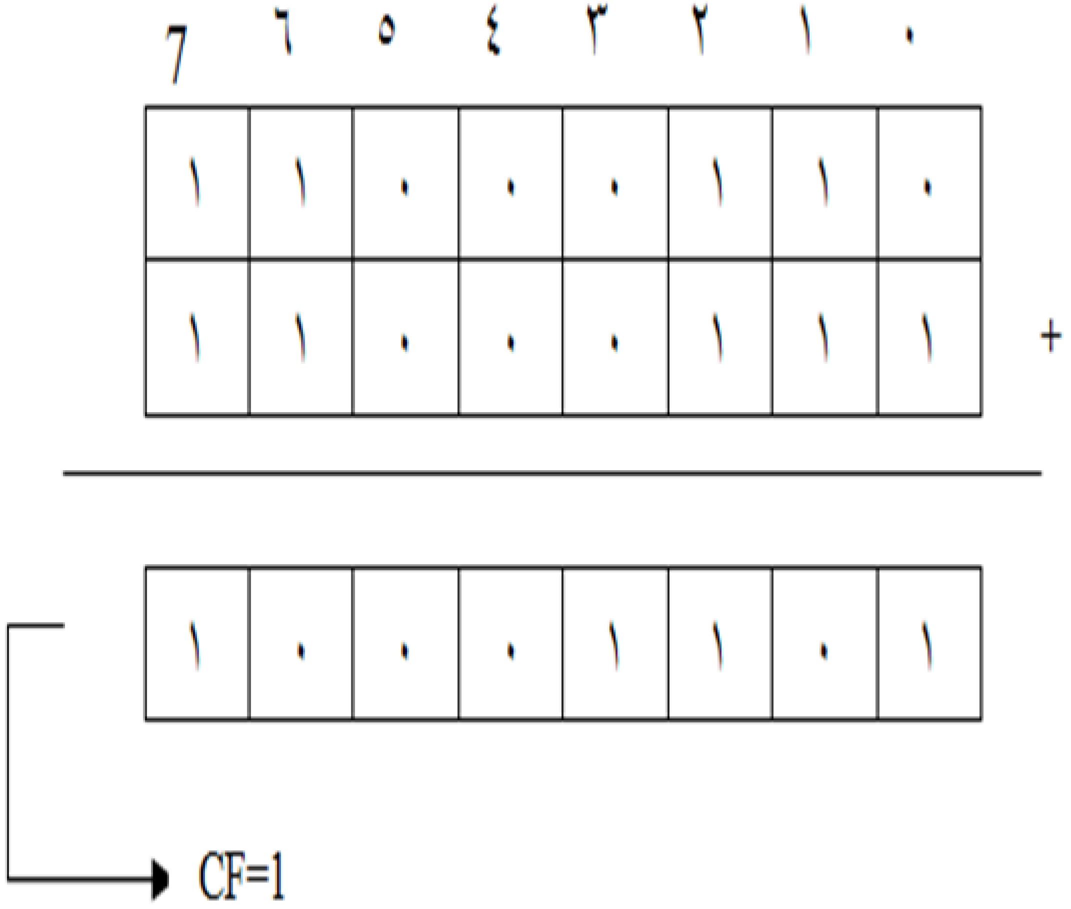
Refers to the resulting situations as a result of the implementation of a logical or mathematical instruction. it either in a single logical (set) or are in the case of a zero logical (reset). We will summarize in the following:

1. Carry Flag

- In the case of the logical one if there is an external displacement (carry) or borrow for the last digit (last bit) and that during the implementation of mathematical instructions.
- In the case of logical zero if there is no carry or borrows for the last decision. Examples:

Examples

First: The Carry



Second: The Borrow

	٧	٦	٥	٤	٣	٢	١	٠	
1	٠	١	٠	١	١	٠	١	١	
	١	١	١	١	١	٠	٠	٠	-
	٠	١	١	٠	٠	٠	١	١	

CF=1

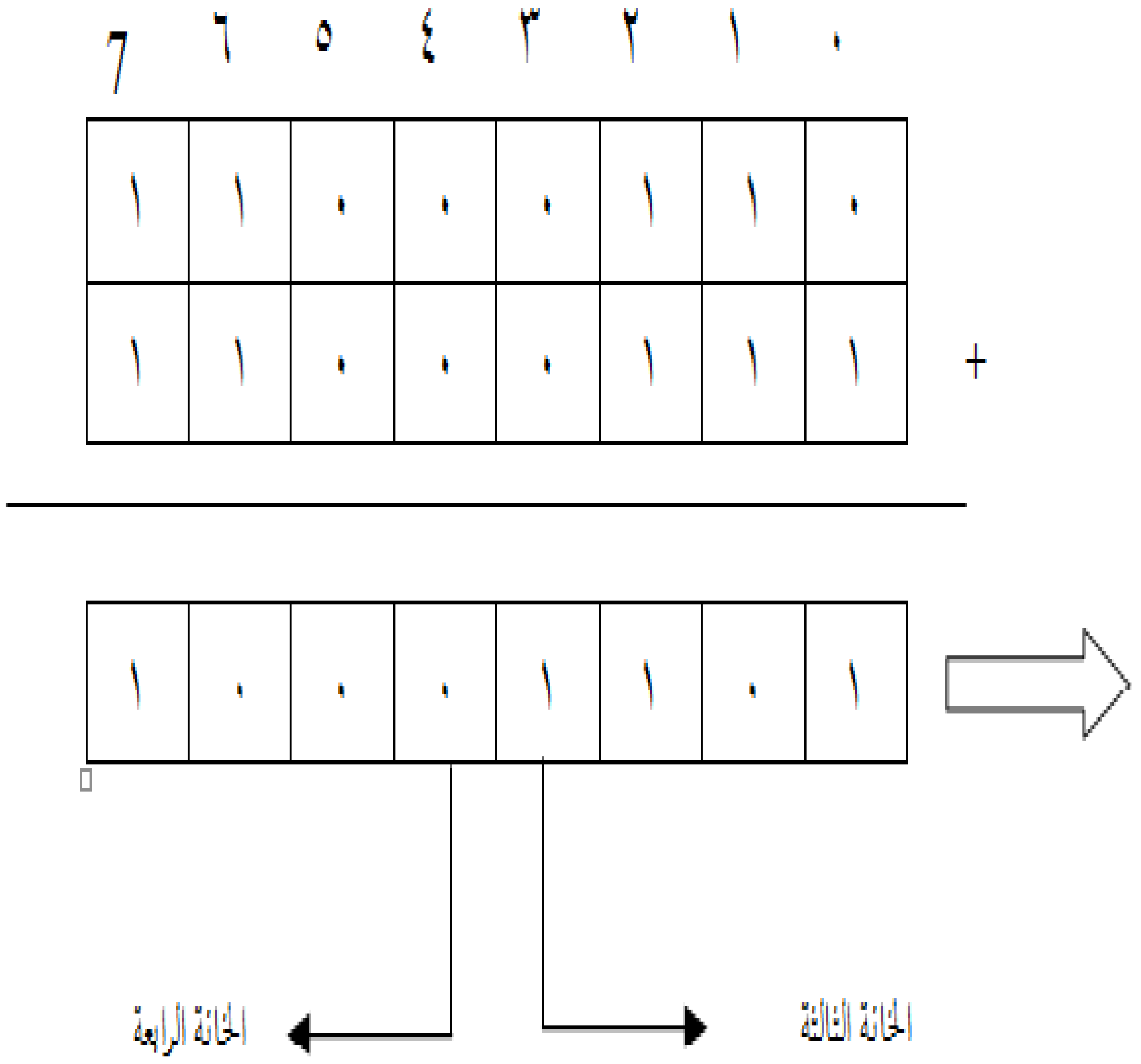
2. Parity Flag PF

- Becomes in one logical case if the last result of an instruction contains even number of ones (after conversion to the binary system of course) otherwise it will be zero logical. Note that the PF flag examines the lower byte only if we are dealing with a word (2 bytes), but when we deal with only one byte, it examines it all.

3- Auxiliary Flag AF

In the case of the logical one if there is a shift from the lower half to the upper half or borrowing from the upper half to the lower half (for the lower byte of the word - 2 bytes). In other words, if we have a displacement from place 3 to place 4, AF = 1 if the data is one byte or two bytes (a word), otherwise AF = 0.

Example:



7	٦	٥	٤	٣	٢	١	٠
١	١	٠	٠	٠	١	١	٠
١	١	٠	٠	٠	١	١	١
١	٠	٠	٠	١	١	٠	١

في هذه الحالة يكون AF=0 لأنه لم يكن معنا باليد واحد عند الانتقال من الخانة الثالثة إلى الخانة الرابعة في الناتج

4- Zero Flag ZF

- It becomes in one logical case when the result of the last calculation or logical operation equals zero.
- It becomes a logical zero if the result of last calculation or logical operation is not equal to zero.

5- Sign Flag SF

- SF flag is in one logical if the result of the last calculation is a negative number.
- SF flag is in the case of a zero logical (reset) if the result of the last calculation is a positive number.

Term: One way to represent negative numbers in the computer is considered as the last place intended to the sign indicates, and as a byte component of eight places the last digit of it cut off in order to sign. it contains one value, the places remaining seven are a binary negative number but if it contains a value Zero, the remaining seven bits are a positive number.

And thus SF is a copy of the final place in output when adopting this system to represent negative numbers. Note that from this principle of representation we can represent the following cases of numbers:

For one byte from -128 to +127

For two bytes from -32768 to +32767

6- Overflow Flag OF

- It is in one logical case if the result does not locate in the storage space, which exceeds the storage capacity. If the result is not outside the specified range, OF remains in the case of logical zero.
- The overflow occurs in the following cases:
  1) add large positive numbers.
  2) add large negative numbers.
  3) Subtract a large positive number from a large negative number.
  4) Subtract a large negative number from a large positive number.
- Note: All previous flags except CF are read-only. We can not change their content so they can only read and their contents can not be changed by direct code.

Here are the control flags:

First: The one step flag (Trap Flag TFA)

it is putting in logical one when we want to execute the program step by step. It is useful when we want to debug our program and troubleshoot places.

Second: Interrupt Flag IF

It is used to express the possibility to perform the interrupt or not. it is putting in one logical case when we do not want to execute any interrupt (interrupt is blocked), and when put in the logical zero, the interrupt is allowed.

Note: An interrupt is a service that leads to a particular action, for Example, interrupt 21, services to return to the operating system.

Third: Direction Flag DF

- Indicates the direction of the sequence of the operations.
- When it in one logical case, the sequence will be from the top (heading) to the lower address.
- When it is in the case of a logical zero, the sequence will be from the lower address to the top address.

The structure of the memory

- Memory consists of a set sequential rooms (8-bit or 1-byte).
- These rooms are numbered from zero to the end of memory. The hexadecimal system is usually used in the numbering process, so each cell has a number that distinguishes it from another.
- The data stored in each room is called the content.

- There are two buses between the processor and the memory, namely the 16-bit data bus and the 20-bit address bus.
- For example, when the processor needs the value stored in the cell 100, the number 100 is binary, placed on the address bar and sent to memory. Once the memory receives this address, the contents of room 100 are sent to the processor via the data bus.

- The 20-bit address bus (20 transmission lines) means that it can move a 20-digit binary number, which means that the largest value to be placed on the address bus: So the 8086 processor can only address one megabyte of memory.

Memory segments (this section is closely related to registers)

- The processor deals, as we mentioned, with one megabyte of memory, and we can divide this mega into four basic segments, that our program deals directly with it (ie, not all memory is used simultaneously).
- These four segments are:

1) Code Segment CS

- This segment allocates memory as it is clear from its name to store the program code.
- A registrar with the same CS name in the processor contains a value indicating to the beginning of this segment in memory and is assisted by the IP (Instruction Pointer) registrar, which keeps the address of the instruction that will be executed now and automatically modifies its value to indicate the address of the following instruction.

2- Data Segment DS

- This segment allocates memory to store data and variables.
- A register with the same name DS in the processor contains a value indicating to the beginning of this segment in memory and is assisted by the SI register that indicates the displacement or movements for its beginning.

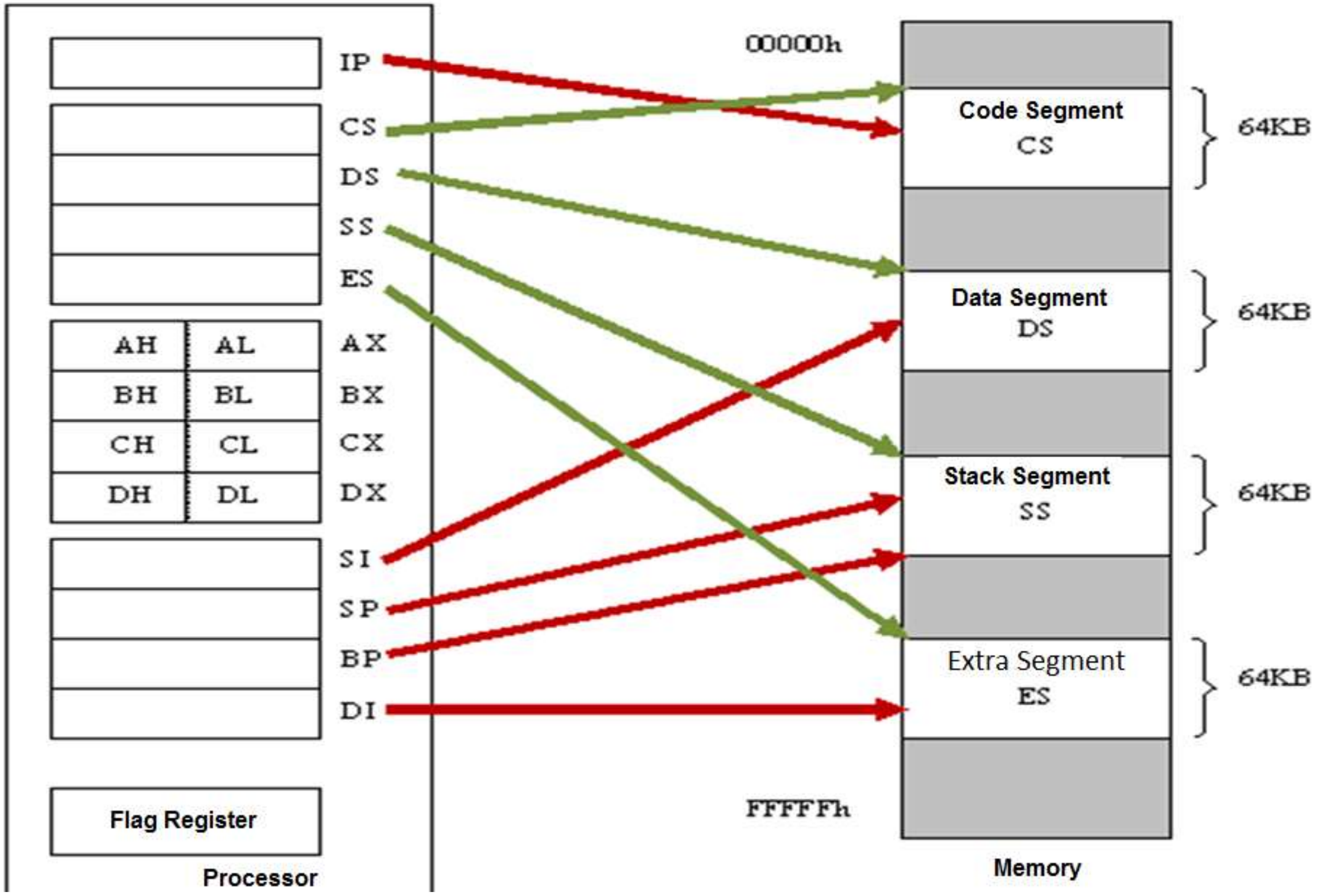
Stack Segment SS

- This segment allocates some necessary information that is feared to be lost or changed during the execution of a program.
- There is a register (SS) has the same name exists in the wizard contains a value indicating the beginning of this segment in memory.

Extra Segment ES

- Used when we need to use two data segments at the same time, so we can take advantage of more memory space.
- It is assisted by the Destination Index DI register in the processor, which indicates the displacement for its start.
- NOTE: we must distinguish between the segment and the segment register where the segment is part of memory while the segment register consists of two bytes and is located in the processor.

The programming model for the 8086



The concept of physical address and displacements

- Introduction

Note that the memory is 1 MB long, that is, numbered from 00000h to FFFFFh. Therefore, we need to label the segments to a 20-bit hex number because a five-digit hexadecimal number (which is used in the numbering of memory cells) needs twenty bits The size of the registers we use in the label are only 16 bits long, forcing us to deduce a 20-bit physical address !!

The mechanism of obtaining the Physical Address PA

To find the physical address, we need two values:

1. the value of the segment register
2. the value of the assist register

So, the physical address is obtained in the following method:

- We take the value of the segment register represented by the hexadecimal system and multiply it by ten hexadecimal systems.
- We collect the value of the assistant register for the same segment and also the hexadecimal system. The result is that we get the physical address
- PA =(segment register x 10h) + Assistant register value

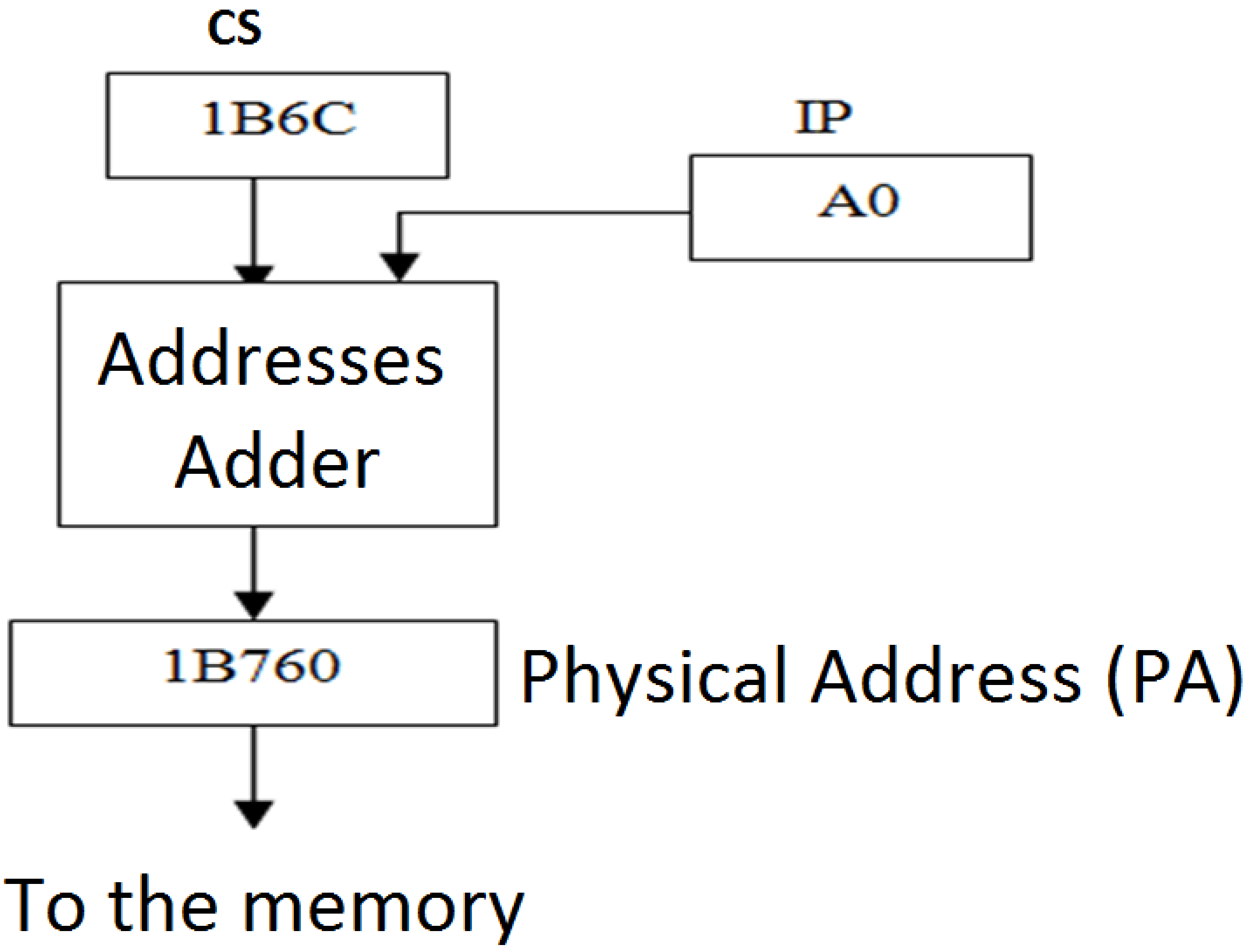
Examples

Suppose, we have a CS code segment register containing 1B6C and the value of the instruction pointer (IP)register is A0.

Find the Physical Address of the instruction.

The solution:

PA = (CS x 10h) + IP = 1B6C x 10h + A0 = 1B760


CS
1B6C
IP
A0
Addresses Adder
1B760
Physical Address (PA)
To the memory

Another example:

Find PA with DS = 1000h and SI = 1F.

The solution:

PA = (1000 x 10) + 1F = 1001F

Inverse method

When we give the physical address and want to derive the value of the segment's register (segment title) and the value of its assistant (helper) register, we follow one of the following methods:

The first method

- Take the four right-hand cells from the given physical address and consider it as a displacement (the value of the assistant register).
- Change the first four digits of the produced physical address with zeros.
- Delete the first zero of the resulting number and then produced hexadecimal number is the value of the segment register.

Example:

Assuming we have a number in physical address 41000h, find the value of the DS and the value of its SI.

Solution:

By taking the first four digits from the right, then the SI value is equal to 1000h, and the offset is DS = 4000h

The second method

- Take the first line of the physical address and consider it an offset or the displacement (Assistant register).
- Delete that block from the physical address, so the resulting number is four digits and represents the value of the segment's register.

Example:

Assume PA = 41000h

Solution:

Take the first place SI = 0DS = 4100

We took the remaining digits of the number