Microprocessors

(Intel 8086)

Lecture 3

Outline

- **8086 architecture and features**
- **8086 registers and memory segmentation**

8086 features

- 16-bit Arithmetic Logic Unit
- 16-bit data bus
- 20-bit address bus – 1,048,576 = 1 meg
- 16 I/O lines so it can access 64K I/O ports
- 16 bit flag
- It has 14 -16 bit registers
- Clock frequency range is 5-10 MHZ
- Designed by Intel
- Rich set of instructions
- 40 Pin DIP, Operates in two modes

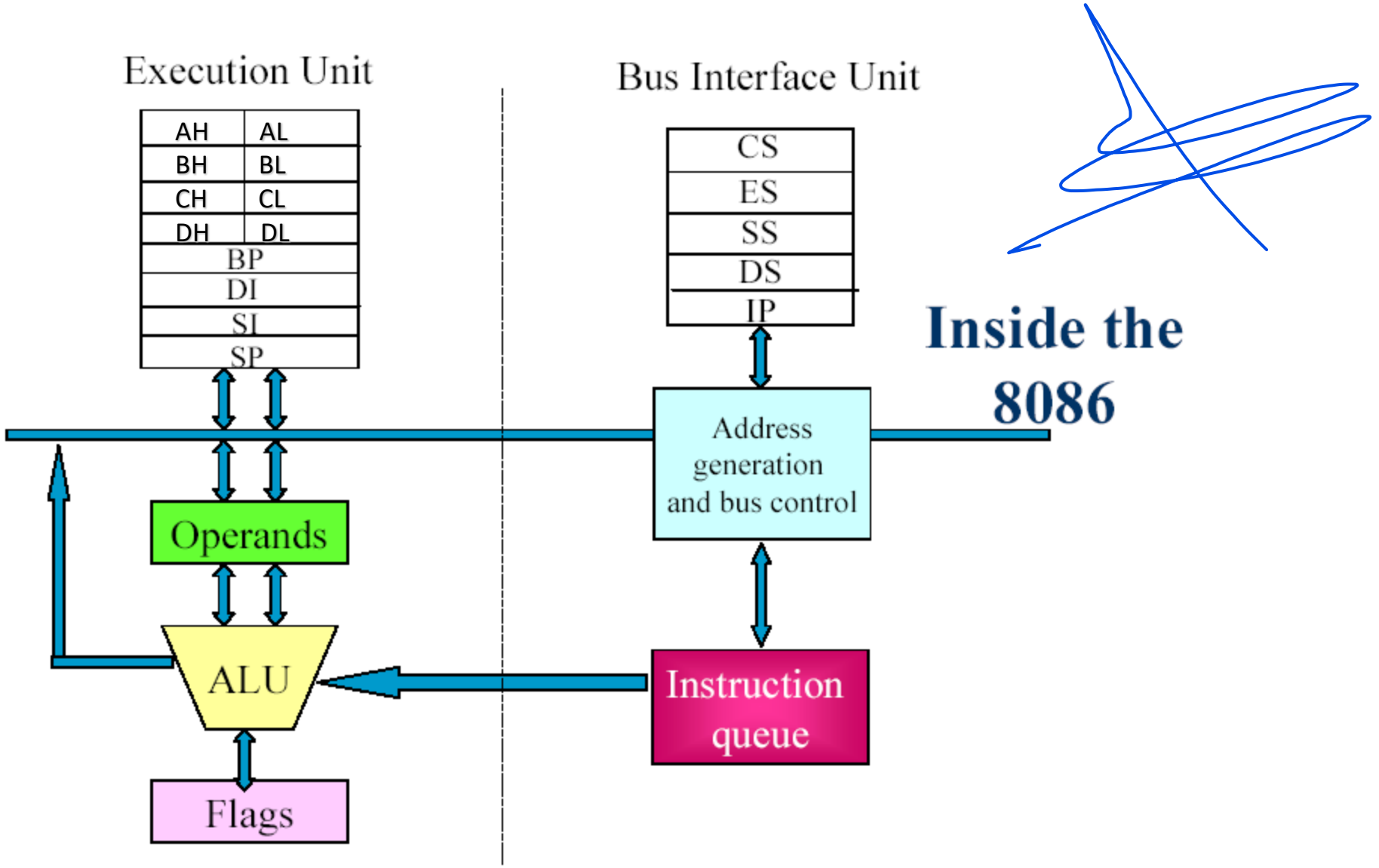
8086 features

- The address refers to a byte in memory. In the 8086, bytes at even addresses come in on the low half of the data bus (bits 0-7) and bytes at odd addresses come in on the upper half of the data bus (bits 8-15).
- The 8086 can read a 16-bit word at an even address in one operation and at an odd address in two operations.

8086 features

- The 8086 has two parts, the Bus Interface Unit (BIU) and the Execution Unit (EU).
- The BIU fetches instructions, reads and writes data, and computes the 20-bit address.
- The EU decodes and executes the instructions using the 16-bit ALU.

Inside 8086


Execution Unit
Bus Interface Unit
AH
AL
BH
BL
CH
CL
DH
DL
BP
DI
SI
SP
CS
ES
SS
DS
IP
Address generation and bus control
Operands
ALU
Instruction queue
Flags
Inside the 8086

Bus Interface Unit (BIU)

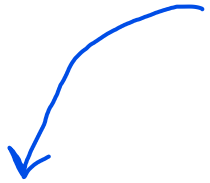
- The BIU fetches instructions using the CS and IP, written CS: IP, to construct the 20-bit address.
- Data is fetched using a segment register (usually the DS) and an effective address (EA) computed by the EU depending on the addressing mode.

Bus Interface Unit (BIU)

The BIU contains the following registers:

- IP - the Instruction Pointer
- CS - the Code Segment Register
- DS - the Data Segment Register
- SS - the Stack Segment Register
- ES - the Extra Segment Register

Execution Unit (EU)

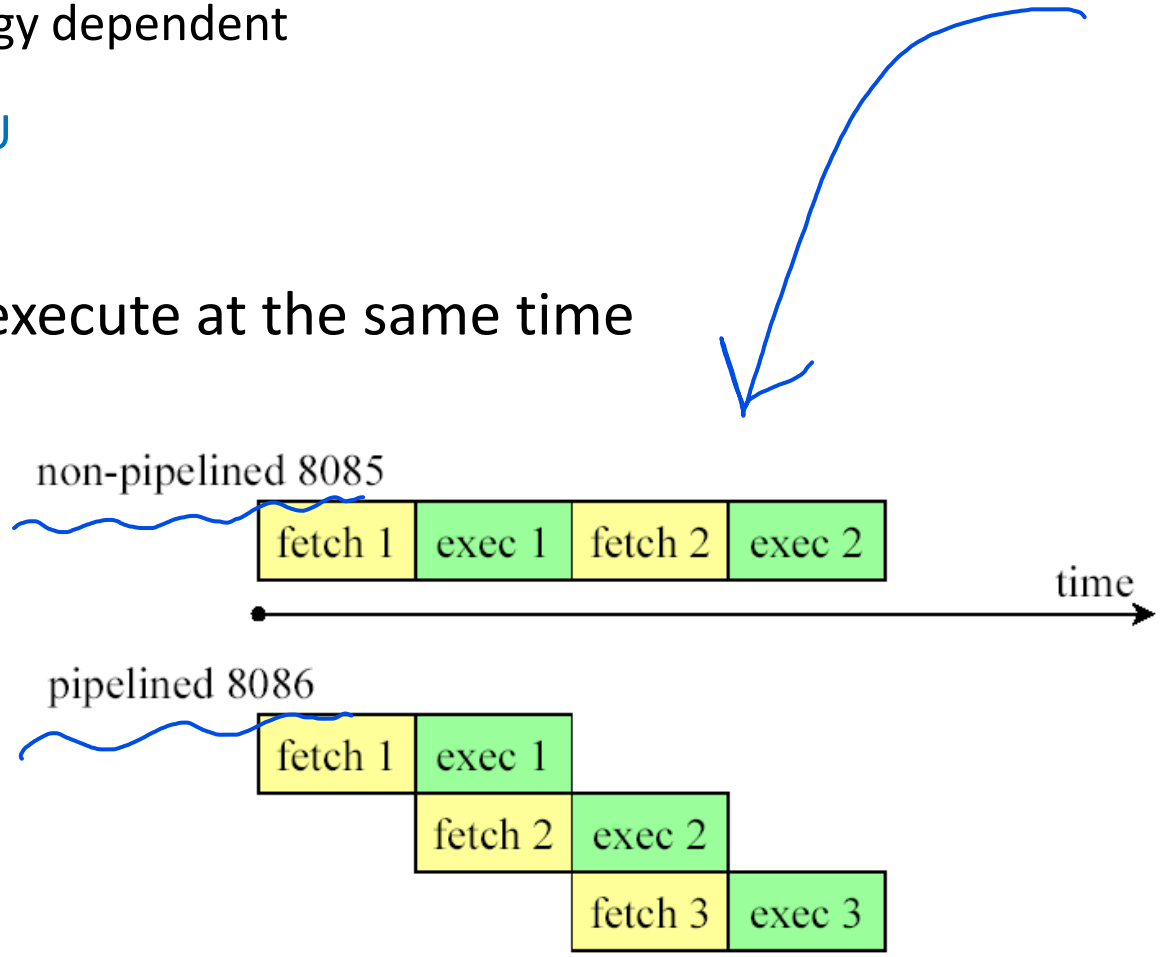


The EU decodes and executes the instructions using the 16-bit ALU.

- AX - the Accumulator
- BX - the Base Register
- CX - the Count Register
- DX - the Data Register
- SP - the Stack Pointer
- BP - the Base Pointer
- SI - the Source Index Register
- DI - the Destination Register

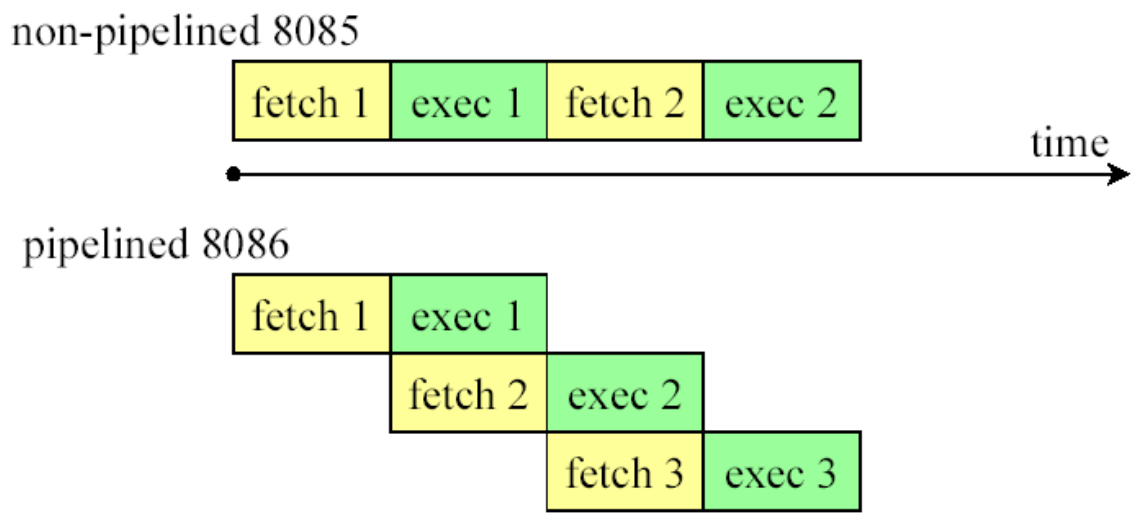
Pipelining

- Two ways to make CPU process information faster:
  - Increase the working frequency – technology dependent
  - Change the internal architecture of the CPU
- Pipelining is to allow CPU to fetch and execute at the same time



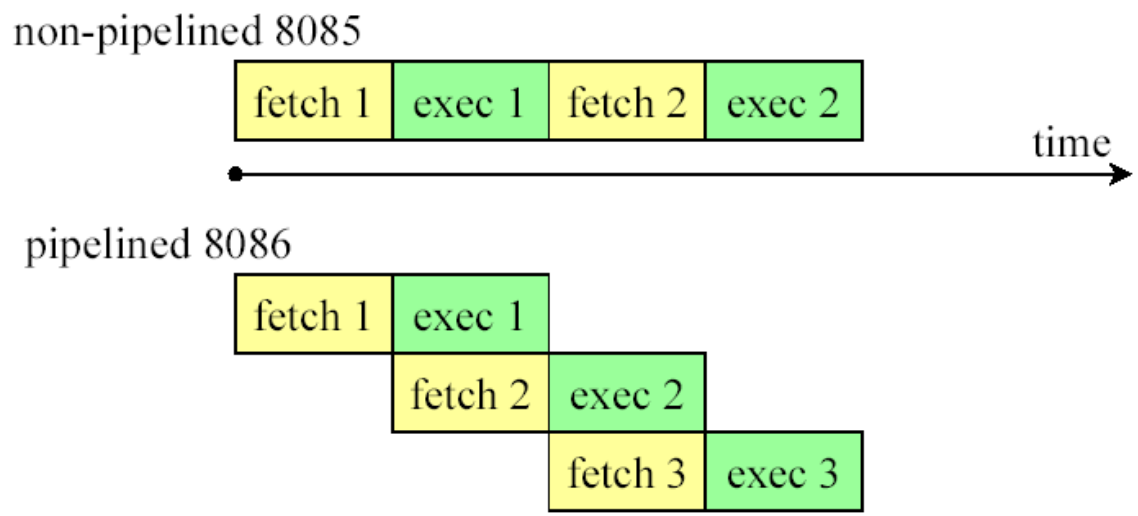
8086 Has Pipelining Architecture:

- While the EU is decoding an instruction or executing an instruction, which does not require use of the buses, the BIU fetches up to six instruction bytes for the following instructions.
- The BIU stores these pre-fetched bytes in a first-in-first-out register set called a **queue.**
- When the EU is ready for its next instruction from the queue in the BIU. This is much faster than sending out an address to the system



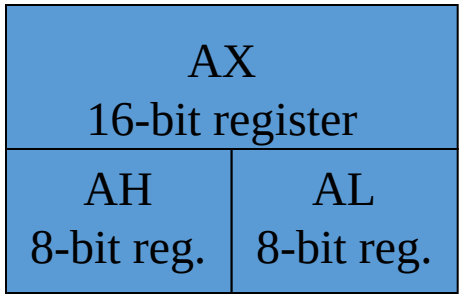
memory and waiting for memory to send back the next instruction byte or bytes.

- Except in the case of JMP and CALL instructions, where the queue must be dumped and then reloaded starting from a new address, this pre-fetch and queue scheme greatly speeds up processing.
- Fetching the next instruction while the current instruction executes is called **pipelining**.

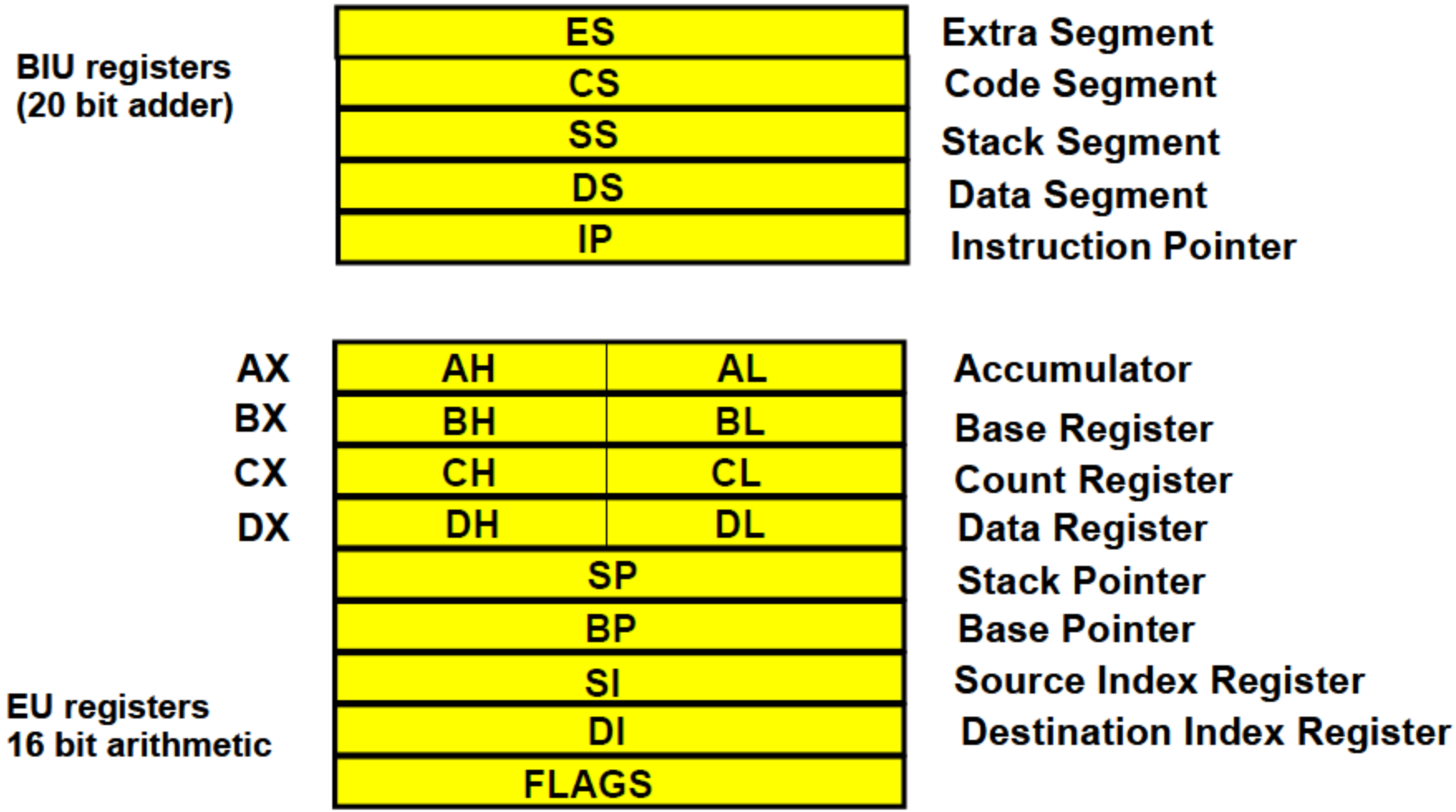


Registers

- To store information temporarily



Category	Bits	Register Names
General	16	AX, BX, CX, DX
	8	AH, AL, BH, BL, CH, CL, DH, DL
Pointer	16	SP (stack pointer), BP (base pointer)
Index	16	SI (source index), DI (destination index)
Segment	16	CS (code segment), DS (data segment)
		SS (stack segment), ES (extra segment)
Instruction	16	IP (instruction pointer)
Flag	16	FR (flag register)


BIU registers
(20 bit adder)
ES
CS
SS
DS
IP
Extra Segment
Code Segment
Stack Segment
Data Segment
Instruction Pointer
AX
BX
CX
DX
AH
AL
BH
BL
CH
CL
DH
DL
SP
BP
SI
DI
FLAGS
Accumulator
Base Register
Count Register
Data Register
Stack Pointer
Base Pointer
Source Index Register
Destination Index Register
EU registers
16 bit arithmetic

Register names

- Accumulator
- Base index
- Count
- Data
- Stack Pointer
- Base Pointer
- Destination index
- Source index
- Instruction Pointer
- Flags

Segment registers

- Code
- Data
- Extra
- Stack

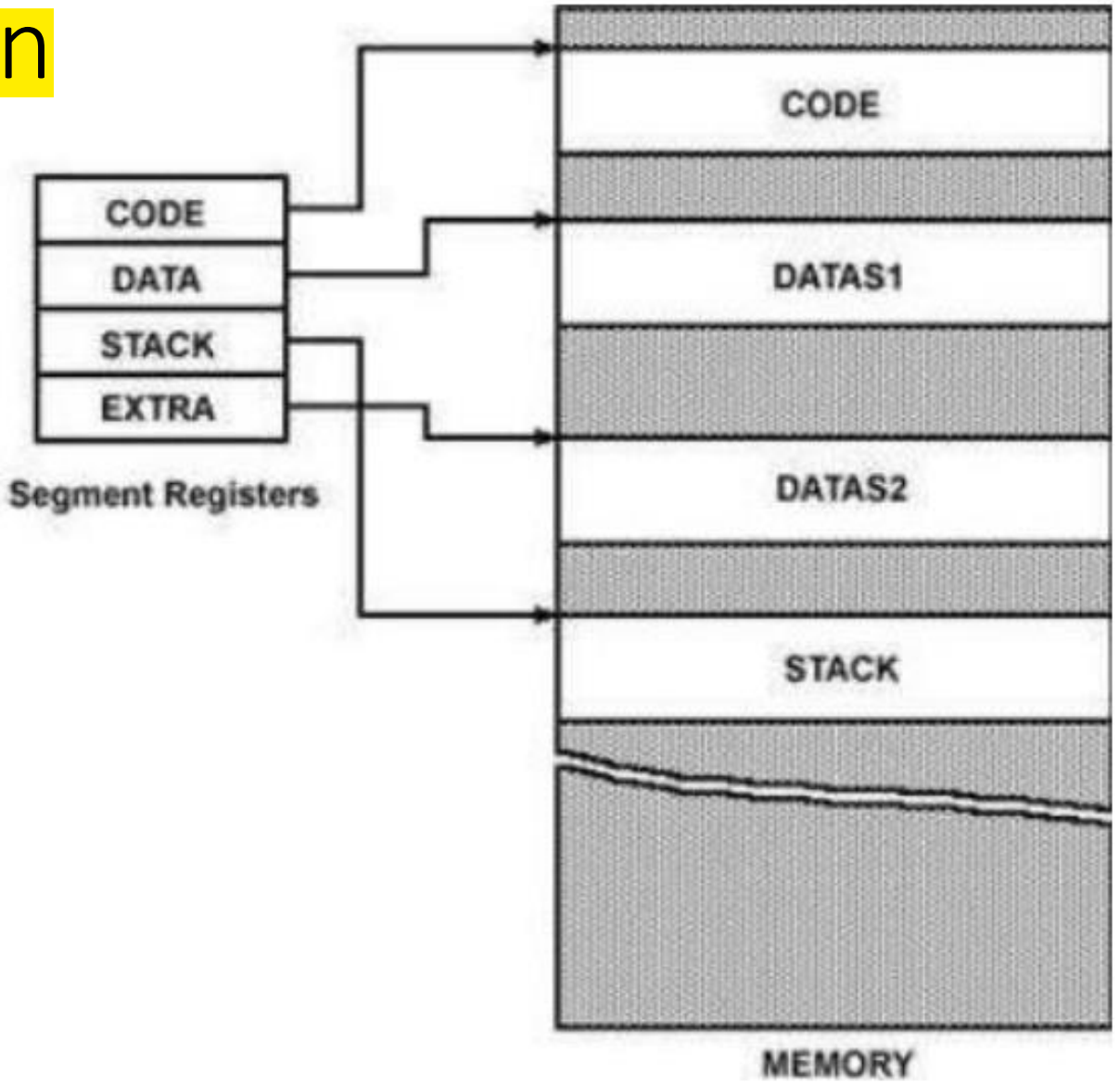
Memory segmentation

- The memory in an 8086 based system is organized as segmented memory.
- The CPU 8086 is able to access 1MB of physical memory. The complete 1MB of memory can be divided into 16 segments, each of 64KB size and is addressed by one of the segment register.
- The 16-bit contents of the segment register actually point to the starting location of a particular segment.
- The address of the segments may be assigned as 0000H to F000h respectively.

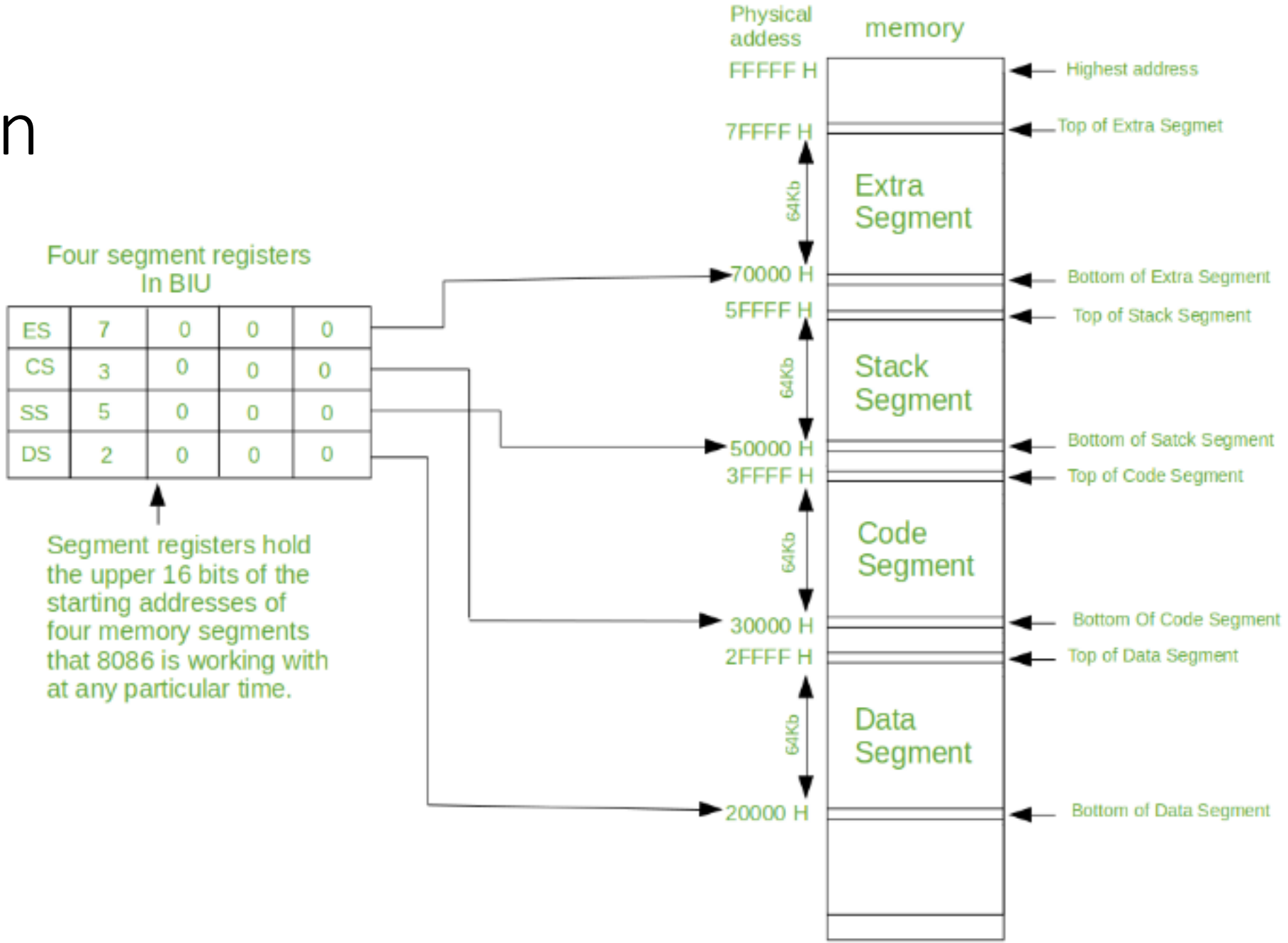
Memory segmentation

- To address a specific memory location within a segment, we need an offset address. The offset address values are from 0000H to FFFFH so that the physical addresses range from 00000H to FFFFFH.
- A program can have more than four segments, but can only access four segments at a time.

Memory segmentation



Memory segmentation



Physical address calculations

Ex:

Segment address =1005H

Offset address =5555H

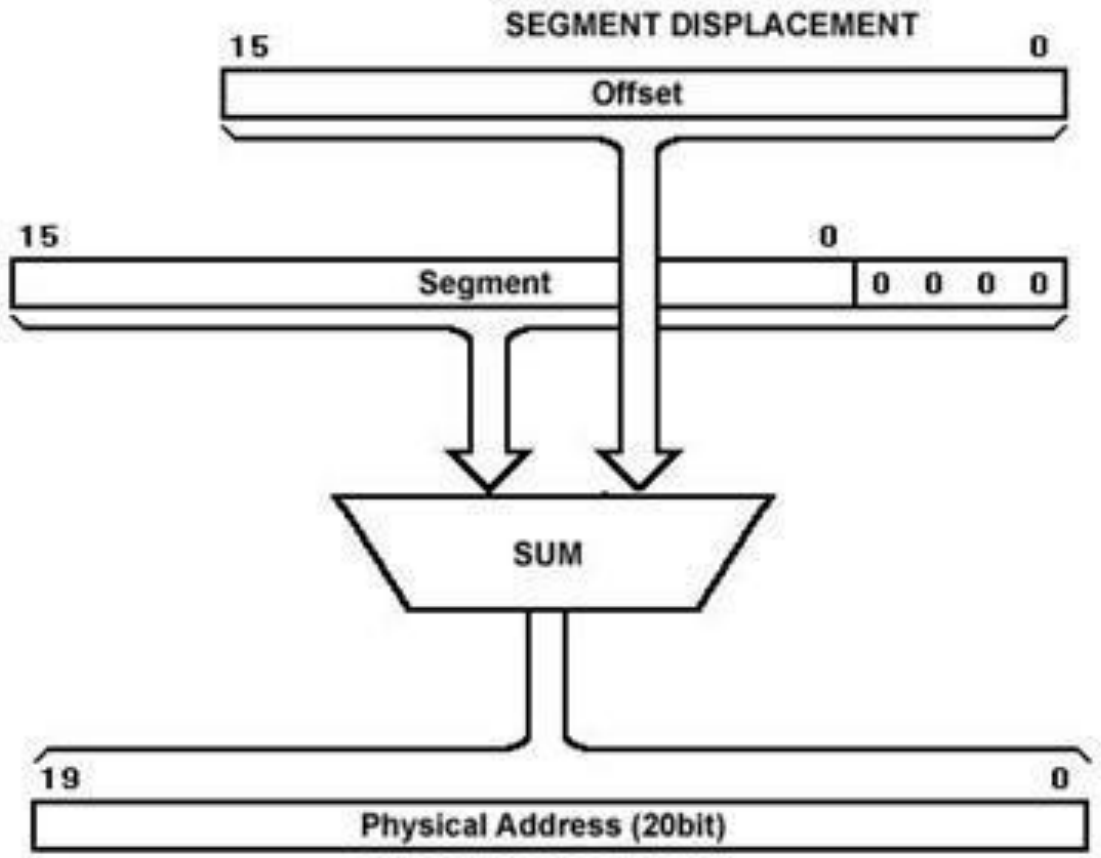
Segment address =1005H = 0001 0000 0000 0101

Shifted left by 4 Positions=0001 0000 0000 0101 0000 + Offset address (= 5555H= 0101 0101 0101 0101)

Physical address=155A5H =0001 0101 0101 1010 0101

Physical address = Segment address * 10H + Offset address.

Physical address calculations



Physical address calculations

Example:

we have segment no 6020h and offset is 4267h then 60200+4267=64467h is the physical address

General Purpose Registers

- **Accumulator register**: consists of two 8-bit registers AL and AH, which can be combined together and used as a 16-bit register AX.
- AL in this case contains the low order byte of the word, and
- AH contains the high order byte.
- Accumulator can be used for I/O operations and string manipulation.

General Purpose Registers

- **Base register**: consists of two 8-bit registers BL and BH, which can be combined together and used as a 16-bit register BX.
- BL in this case contains the low order byte of the word, and BH contains the high order byte.
- BX register usually contains a data pointer used for based, based indexed or register indirect addressing.

General Purpose Registers

- **Count register:** consists of two 8-bit registers CL and CH, which can be combined together and used as a 16-bit register CX.
- CL in this case contains the low order byte of the word, and CH contains the high order byte.
- Count register can be used in Loop, shift/rotate instructions and as a counter in string manipulation.

General Purpose Registers

- **Data register**: consists of two 8-bit registers DL and DH, which can be combined together and used as a 16-bit register DX.
- DL in this case contains the low order byte of the word, and DH contains the high order byte.
- Data register can be used as a port number in I/O operations. In integer 32-bit multiply and divide instruction the DX register contains high order word of the resulting number.

Index and Pointer Register

- **Source Index (SI):** is a 16-bit register. SI is used for indexed, based indexed and register indirect addressing.
- Source data address in string manipulation instructions. Used in conjunction with DS register to point to data locations in the data segment.

Index and Pointer Register

- **Destination Index (DI)** is a 16-bit register. Used with the ES register in string operations. DI is used for indexed, based indexed and register indirect addressing.
- A destination data address in string manipulation instructions.
- Destination Index and SI Source Index registers are used to hold address.

Index and Pointer Register

- **Stack Pointer (SP)**: is a 16-bit register pointing to program stack, it is used to hold the address of the top of the stack.
- The stack is maintained as LIFO with its bottom at the start of the stack segment (Specified by the SS segment register).
- Unlike the SP register, the BP can be used to specify the offset of other program segments.

Index and Pointer Register

- **Base Pointer (BP):** is a 16-bit register pointing to program stack segment.
- It is usually used by subroutine to locate variables that were passed on stack by calling program.
- BP register is usually used for based, based indexed or register indirect addressing.

Segment Registers

- **Code Segment (CS)** is a 16-bit register containing address of 64 KB segment with processor instructions.
- The processor uses CS segment for all accesses to instructions referenced by instruction pointer (IP) register.
- CS register cannot be changed directly. The CS register is automatically updated during far jump, far call and far return instructions.

Segment Registers

- **Stack segment (SS)** is a 16-bit register containing address of 64KB segment with program stack.
- The processor assumes that all data referenced by the stack pointer (SP) and base pointer (BP) registers is located in the stack segment.
- SS register can be changed directly using POP instruction.

Segment Registers

- **Data segment (DS)** is a 16-bit register containing address of 64KB segment with program data.
- The processor assumes that all data referenced by general registers (AX, BX, CX, DX) and index register (SI, DI) is located in the data segment.
- DS register can be changed directly using POP and LDS instructions.

Segment Registers

- **Extra segment (ES)** used to hold the starting address of Extra segment.
- Extra segment is provided for programs that need to access a second data segment.
- Segment registers cannot be used in arithmetic operations.

Instruction Pointer (IP)

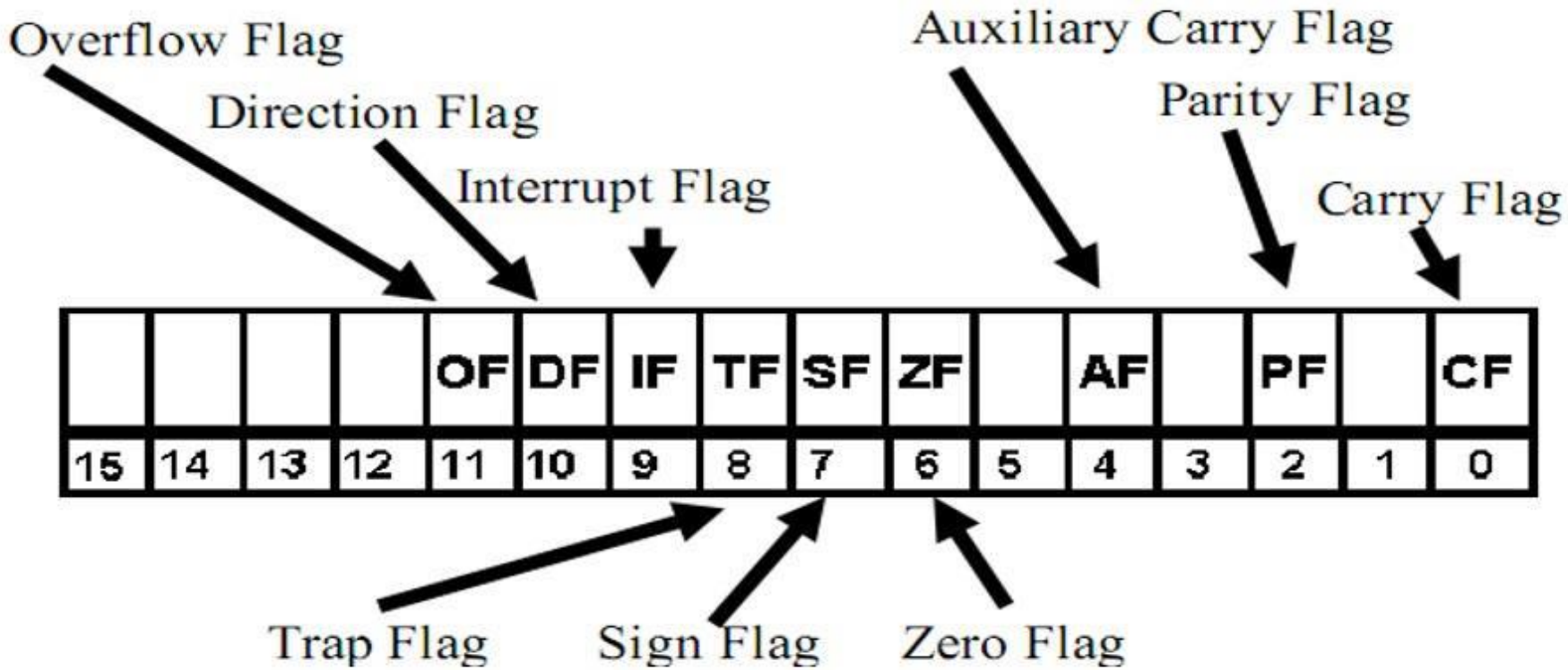
- It is a 16-bit register.
- It is used to control which instruction the CPU executes.
- The IP, or program counter, is used to store the memory location of the next instruction to be executed.
- The CPU checks the program counter to ascertain which instruction to carry out next. It then updates the program counter to point to the next instruction.

Flag Register

- It determines the current state of the processor. They are modified automatically by CPU after mathematical operations, this allows to determine the type of the result, and to determine conditions to transfer control to other parts of the program. 8086 has 9 flags and they are divided into two categories:

  - Status flag
  - Control flag

Flag Register



Status Flags

Status Flags represent result of last arithmetic or logical instruction executed. Conditional flags are as follows:

- **Carry Flag (CF):** This flag indicates an overflow condition for unsigned integer arithmetic. It is also used in multiple-precision arithmetic.

- **Parity Flag (PF):** This flag is used to indicate the parity of result. If lower order 8-bits of the result contains even number of 1‟s, the Parity Flag is set and for odd number of 1‟s, the Parity Flag is reset.

Status Flags

- **Zero Flag (ZF):** It is set; if the result of arithmetic or logical operation is zero else it is reset.

- **Sign Flag (SF):** In sign magnitude format the sign of number is indicated by MSB bit. If the result of operation is negative, sign flag is set.

- **Overflow Flag (OF):** It occurs when signed numbers are added or subtracted. An OF indicates that the result has exceeded the capacity of machine.

Thanks