Microprocessors Memory Map

Outline of the Lecture
- **Memory Map of the IBM PC**
- **Pushing and Popping Operations (Stack)**
- **Flag Registers and bit fields**

⚜ MEMORY MAP OF THE IBM PC

➢ The 20-bit address of the 8086/8088 allows 1M byte of (1024 K bytes) memory space with the address range 00000-FFFFF.
➢ The allocation of the memory is called a ***memory map***.

RAM 640K	00000H
	9FFFFH
Video Display RAM 128K	A0000H
	BFFFFH
ROM 256K	C0000H
	FFFFFH

RAM: Memory locations from 00000H to 9FFFFH (640K) are set aside for RAM. In an IBM PC the DOS operating system first allocates the available RAM on the PC for its own use and let the rest be used for applications such as word processors.

The amount of memory used by DOS varies among its various versions. That is why we do not assign any values for the CS, DS, SS, and ES. Memory management functions within DOS handle this for the operating system.

Video RAM: Memory locations from A0000H to BFFFFH (128K) are set aside for video. This amount varies depending on the video board installed on the PC.

ROM: Memory locations from C0000H to FFFFFH (256K) are set aside for ROM. First 64 K bytes is used by BIOS (Basic Input/Output System) ROM. Some of the remaining space is used for adapter cards.
Function of BIOS ROM:
- CPU can only execute programs that are stored in memory, there must be some permanent (nonvolatile) memory to hold the programs telling the CPU what to do when the power is turned on.
- BIOS contains programs to test RAM and other components connected to the CPU.

♣ STACK: PUSHING AND POPPING OPERATIONS

What is a stack, and why is it needed?
The stack is a section of read/write memory (RAM) used by the CPU to store information *temporarily*. CPU needs this storage area since there are *only limited number* of registers.

How stacks are accessed
➢ **SS** (stack segment) and **SP** (stack pointer) must be loaded to access stack in the memory.
➢ Every register in the CPU (except segment registers and SP) can be stored in the stack and loaded from the stack.
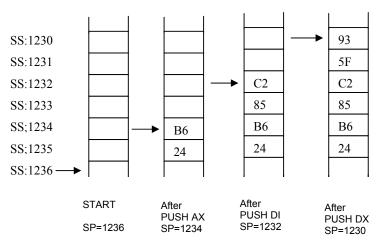
Pushing onto the stack
➢ Storing the CPU register in the stack is called a *push*.

Ex: SP=1236, AX=24B6, DI=85C2, and DX=5F93, show the contents of the stack as each instruction is executed.

PUSH AX
PUSH DI
PUSH DX

Solution:

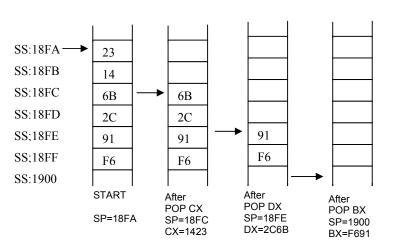|	START SP=1236	After PUSH AX SP=1234	After PUSH DI SP=1232	After PUSH DX SP=1230
SS:1230				93
SS:1231				5F
SS:1232			C2	C2
SS:1233			85	85
SS;1234		B6	B6	B6
SS;1235		24	24	24
SS:1236				

• Note that in 80x86 the lower byte of the register is stored to the lower address.

Popping the stack
➢ Loading the contents of the stack into the CPU register is called a *pop*.

Ex: assume that the stack is shown below, and SP=18FA, show the contents of the stack and registers as each of the following instructions is executed.

POP   CX
POP   DX
POP   BX

Solution:

	START SP=18FA	After POP CX SP=18FC CX=1423	After POP DX SP=18FE DX=2C6B	After POP BX SP=1900 BX=F691
SS:18FA	23			
SS:18FB	14			
SS:18FC	6B	6B		
SS:18FD	2C	2C		
SS:18FE	91	91	91	
SS;18FF	F6	F6	F6	
SS:1900				

Logical vs. physical address of the stack

➢ Calculating the physical address for the stack, the same principle is applied as was used for the code and data segments. Physical address depends on the value of stack segment (SS) register and the stack pointer (SP).

Ex: If SS=3500H and SP:FFFEH
a) Calculate the physical address: 35000+FFFE = 44FFE
b) Calculate the lower range of the stack: 35000+0000 = 35000
c) Calculate the upper range of the stack segment: 35000+FFFF = 44FFF
d) Show the logical address of the stack: 3500:FFFE

✛ THE FLAG REGISTER (FR) AND BIT FIELDS

➢ The flag register is a 16-bit register sometimes referred as the *status* register. Although the register is 16-bit. Not all the bits are used.
➢ *Conditional flags*: 6 of the flags are called the conditional flags, meaning that they indicate some condition that resulted after an instruction was executed. These 6 are: CF, PF, AF, ZF, SF, and OF.
➢ The 16 bits of the flag registers:

R	R	R	R	OF	DF	IF	TF	SF	ZF	U	AF	U	PF	U	CF

R=	reserved	SF= sign flag
U=	undefined	ZF= zero flag
OF=	overflow flag	AF= auxiliary carry flag
DF=	direction flag	PF= parity flag
IF=	interrupt flag	CF= carry flag
TF=	trap flag	

CF, the Carry Flag: This flag is set whenever there is a carry out, either from d7 after an 8-bit operation, or from d15 after a 16-bit data operation.

PF, the Parity Flag: After certain operations, the parity of the result's low-order byte is checked. If the byte has an even number of 1s, the parity flag is set to 1; otherwise, it is cleared.

AF, the Auxiliary Carry Flag: If there is a carry from d3 to d4 of an operation this bit is set to 1, otherwise cleared (set to 0).

ZF, the Zero Flag: The ZF is set to 1 if the result of the arithmetic or logical operation is zero, otherwise, it is cleared (set to 0).

SF, the Sign Flag: MSB is used as the sign bit of the binary representation of the signed numbers. After arithmetic or logical operations the MSB is copied into SF to indicate the sign of the result.

TF, the Trap Flag: When this flag is set it allows the program to single step, meaning to execute one instruction at a time. Used for debugging purposes.

IF, Interrupt Enable Flag: This bit is set or cleared to enable or disable only the external interrupt requests.

DF, the Direction Flag: This bit is used to control the direction of the string operations.

OF, the Overflow Flag: This flag is set whenever the result of a signed number operation is too large, causing the high-order bit to overflow into the sign bit.