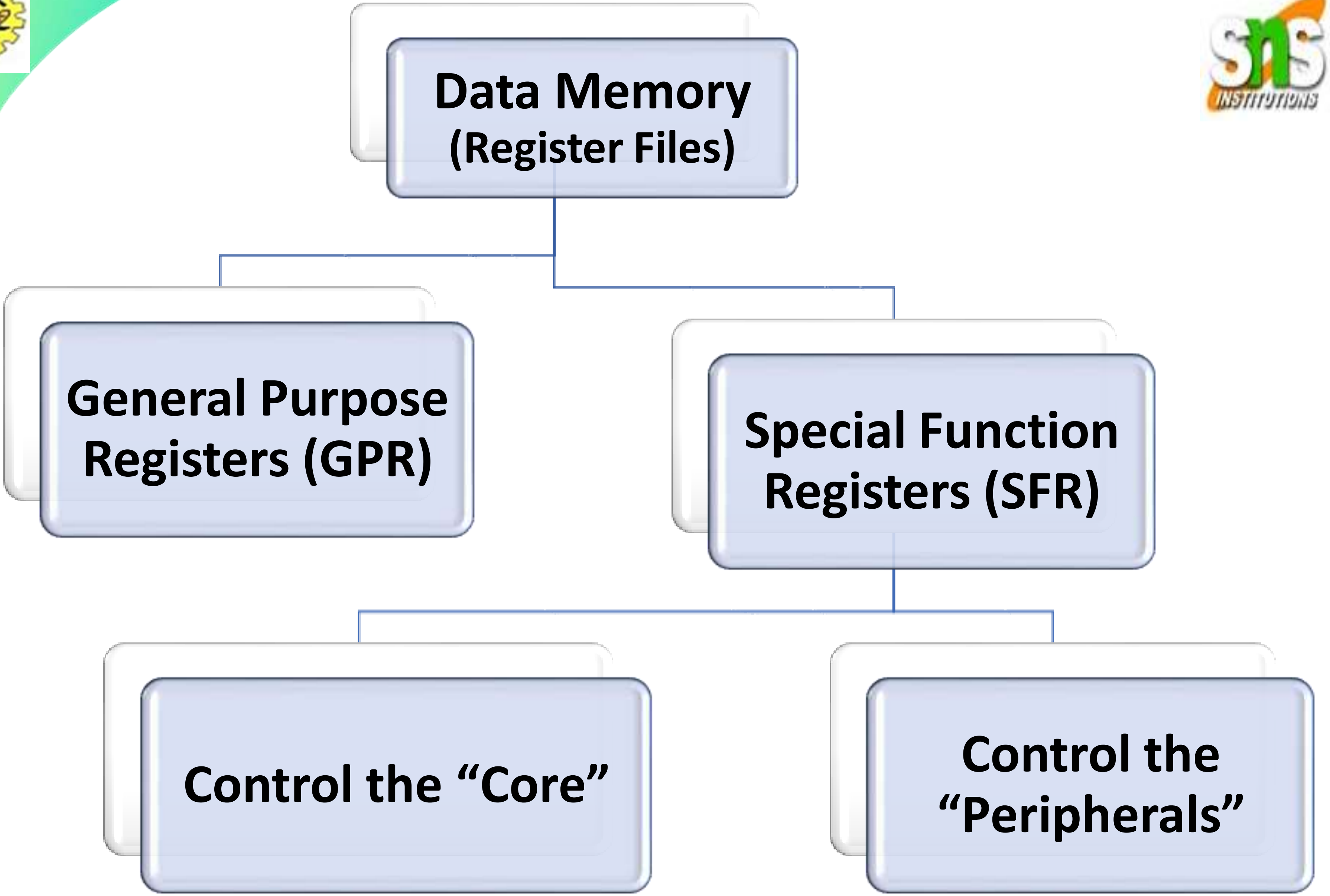
Memory Organization:

It has three memory blocks.

- Program memory
- Data memory
- Stack
- **Program memory :**
- PIC16C7X family has a 13 program counter
- Capable of addressing an **8K x 14** program memory



Data Memory (Register Files)

General Purpose Registers (GPR)

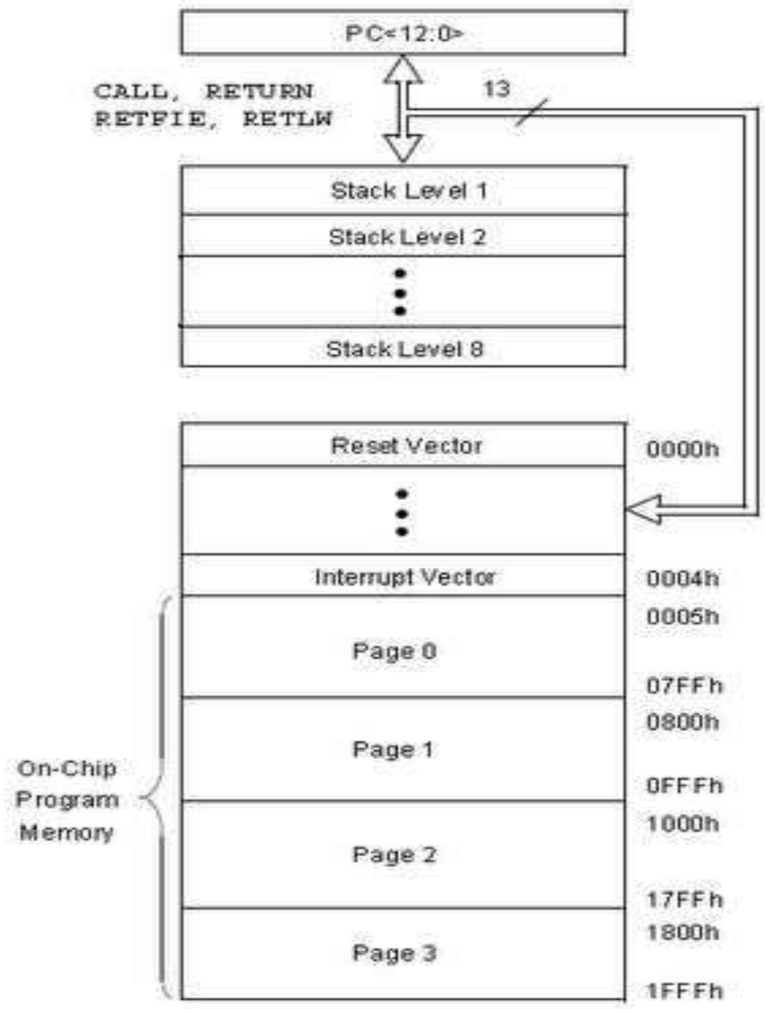
Special Function Registers (SFR)

Control the “Core”

Control the “Peripherals”

- **Program Memory** - A memory that contains the program(which we had written), after we've burned it. As a reminder, Program Counter executes commands stored in the program memory, one after the other.
- **Data Memory** – This is RAM memory type, which contains a special registers like SFR (Special Faction Register) and GPR (General Purpose Register). The variables that we store in the Data Memory during the program are deleted after we turn of the micro.
- These two memories have separated data buses, which makes the access to each one of them very easy.
- **Data EEPROM (Electrically Erasable Programmable Read-Only Memory)** - A memory that allows storing the variables as a result of burning the written program.
- Each one of them has a different role. Program Memory and Data Memory two memories that are needed to build a program, and Data EEPROM is used to save data after the microcontroller is turn off.
  Program Memory and Data EEPROM they are non-volatile memories, which store the information even after the power is turn off. These memories called Flash Or EEPROM. In contrast, Data Memory does not save the information because it needs power in order to maintain the information stored in the chip.

- The PIC16F87XA devices have a 13-bit program counter capable of addressing an 8K word x 14 bit program memory space. This memory is used to store the program after we burn it to the microcontroller.
- The PIC16F876A/877A devices have 8K words x 14 bits of Flash program memory that can be electrically erased and reprogrammed. Each time we burn program into the micro, we erase an old program and write a new one.


PC<12:0>
CALL, RETURN
RETFIE, RETLW
13
Stack Level 1
Stack Level 2
Stack Level 8
Reset Vector
0000h
Interrupt Vector
0004h
0005h
Page 0
07FFh
0800h
Page 1
0FFFh
1000h
Page 2
17FFh
1800h
Page 3
1FFFh
On-Chip
Program
Memory

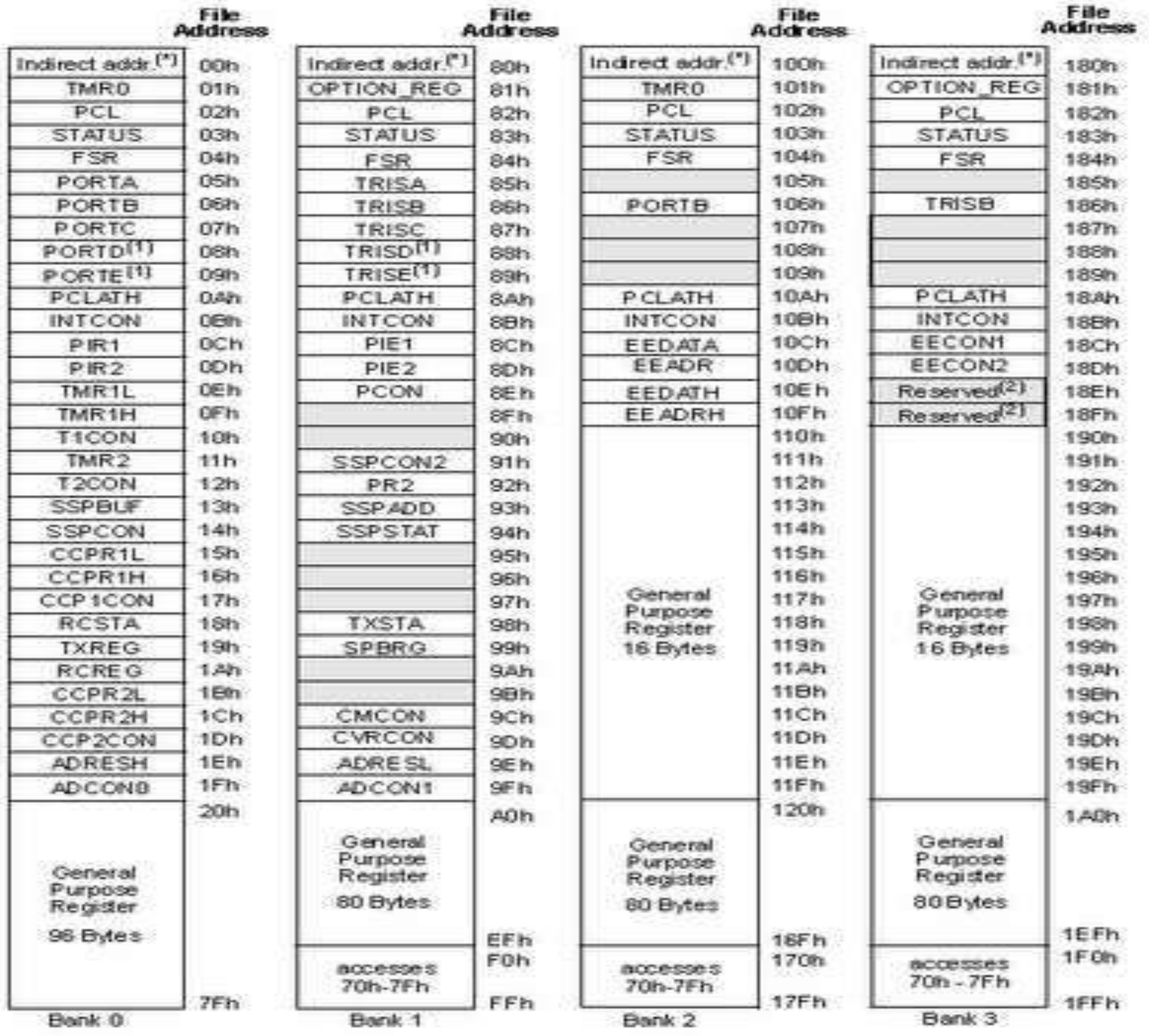
- Program Counter (PC) keeps track of the program execution by holding the address of the current instruction. It is automatically incremented to the next instruction during the current instruction execution.

- The PIC16F87XA family has an 8-level deep x 13-bit wide hardware stack. The stack space is not part of either program or data space and the stack pointer is not readable or writable. In the PIC microcontrollers, this is a special block of RAM memory used only for this purpose.

- The CALL instruction is used to jump to a subroutine, which must be terminated with the RETURN instruction. CALL has the address of the first instruction in the subroutine as its operand. When the CALL instruction is executed, the destination address is copied to the PC. The PC is PUSHed onto the stack when a CALL instruction is executed, or an interrupt causes a branch. The stack is POP’ed in the event of a RETURN, RETLW or a RETFIE instruction execution.
- The stack operates as a circular buffer. This means that after the stack has been PUSHed eight times, the ninth push overwrites the value that was stored from the first push. The tenth push overwrites the second push (and so on).
- Each time the main program execution starts at address 0000 - Reset Vector. The address 0004 is “reserved” for the “interrupt service routine” (ISR).

- **Data Memory Organization**
- The data memory is partitioned into multiple banks which contain the General Purpose Registers and the Special Function Registers. Number of banks may vary depending on the microcontroller; for example, micro PIC16F84 has only two banks.
- Each bank extends up to 7Fh (128 bytes). The lower locations of each bank are reserved for the Special Function Registers. Above the Special Function Registers are General Purpose Registers, implemented as static RAM. While program is being executed, it is working with the particular bank. The default bank is **BANK0.**
- To access a register that is located in another bank, one should access it inside the program. There are special registers which can be accessed from any bank, such as STATUS register.

•



	File Address		File Address		File Address		File Address
Indirect addr.[*]	00h	Indirect addr.[*]	80h	Indirect addr.[*]	100h	Indirect addr.[*]	180h
TMR0	01h	OPTION_REG	81h	TMR0	101h	OPTION_REG	181h
PCL	02h	PCL	82h	PCL	102h	PCL	182h
STATUS	03h	STATUS	83h	STATUS	103h	STATUS	183h
FSR	04h	FSR	84h	FSR	104h	FSR	184h
PORTA	05h	TRISA	85h		105h		185h
PORTB	06h	TRISB	86h	PORTB	106h	TRISB	186h
PORTC	07h	TRISC	87h		107h		187h
PORTD[1]	08h	TRISD[1]	88h		108h		188h
PORTE[1]	09h	TRISE[1]	89h		109h		189h
PCLATH	0Ah	PCLATH	8Ah	PCLATH	10Ah	PCLATH	18Ah
INTCON	0Bh	INTCON	8Bh	INTCON	10Bh	INTCON	18Bh
PIR1	0Ch	PIE1	8Ch	EEDATA	10Ch	EECON1	18Ch
PIR2	0Dh	PIE2	8Dh	EEADR	10Dh	EECON2	18Dh
TMR1L	0Eh	PCON	8Eh	EEDATH	10Eh	Reserved[2]	18Eh
TMR1H	0Fh		8Fh	EEADRH	10Fh	Reserved[2]	18Fh
T1CON	10h		90h	General Purpose Register 16 Bytes	110h	General Purpose Register 16 Bytes	190h
TMR2	11h	SSPCON2	91h		111h		191h
T2CON	12h	PR2	92h		112h		192h
SSPBUF	13h	SSPADD	93h		113h		193h
SSPCON	14h	SSPSTAT	94h		114h		194h
CCPR1L	15h		95h		115h		195h
CCPR1H	16h		96h		116h		196h
CCP1CON	17h		97h		117h		197h
RCSTA	18h	TXSTA	98h		118h		198h
TXREG	19h	SPBRG	99h		119h		199h
RCREG	1Ah		9Ah		11Ah		19Ah
CCPR2L	1Bh		9Bh		11Bh		19Bh
CCPR2H	1Ch	CMCON	9Ch		11Ch		19Ch
CCP2CON	1Dh	CVRCON	9Dh		11Dh		19Dh
ADRESH	1Eh	ADRESL	9Eh		11Eh		19Eh
ADCON0	1Fh	ADCON1	9Fh		11Fh		19Fh
General Purpose Register 96 Bytes	20h	General Purpose Register 80 Bytes	A0h	General Purpose Register 80 Bytes	120h	General Purpose Register 80 Bytes	1A0h
			EFh		16Fh		1EFh
		accesses 70h-7Fh	F0h	accesses 70h-7Fh	170h	accesses 70h - 7Fh	1F0h
	7Fh		FFh		17Fh		1FFh
Bank 0		Bank 1		Bank 2		Bank 3	

- The data EEPROM and Flash program memory is readable and writable during normal operation (over the full VDD range). This memory is not directly mapped in the register file space. Instead, it is indirectly addressed through the Special Function Registers.
- There are six SFRs used to read and write to this memory:

1. EECON1
2. EECON2
3. EEDATA
4. EEDATH
5. EEADR
6. EEADRH

- When interfacing to the data memory block, EEDATA holds the 8-bit data for read/write and EEADR holds the address of the EEPROM location being accessed. These devices have 128 or 256 bytes of data EEPROM (depending on the device), with an address range from 00h to FFh. On devices with 128 bytes, addresses from 80h to FFh are unimplemented.
- A few important points about Data EEPROM memory:
- It lets you save data DURING programming
- The data is saved during the “burning” process
- You can read the data memory during the programming and use it
- The use is made possible with the help of SFR
- At this point there is no need to learn how to use this memory with special registers, because there are functions (writing and reading) that are ready.