Embedded Systems Design

(630470)

Lecture 4

Memory Organization

Prof. Kasim M. Al-Aubidy

Computer Eng. Dept.

Memory Organization:

AVAILABLE MEMORY IN PIC

Device	Program Flash	Data Memory	Data EEPROM
PIC16F87/88	4K x 14	368 x 8	256 x 8
PIC16F84	**1K x 14**	**128 x 8**	**64 x 8**

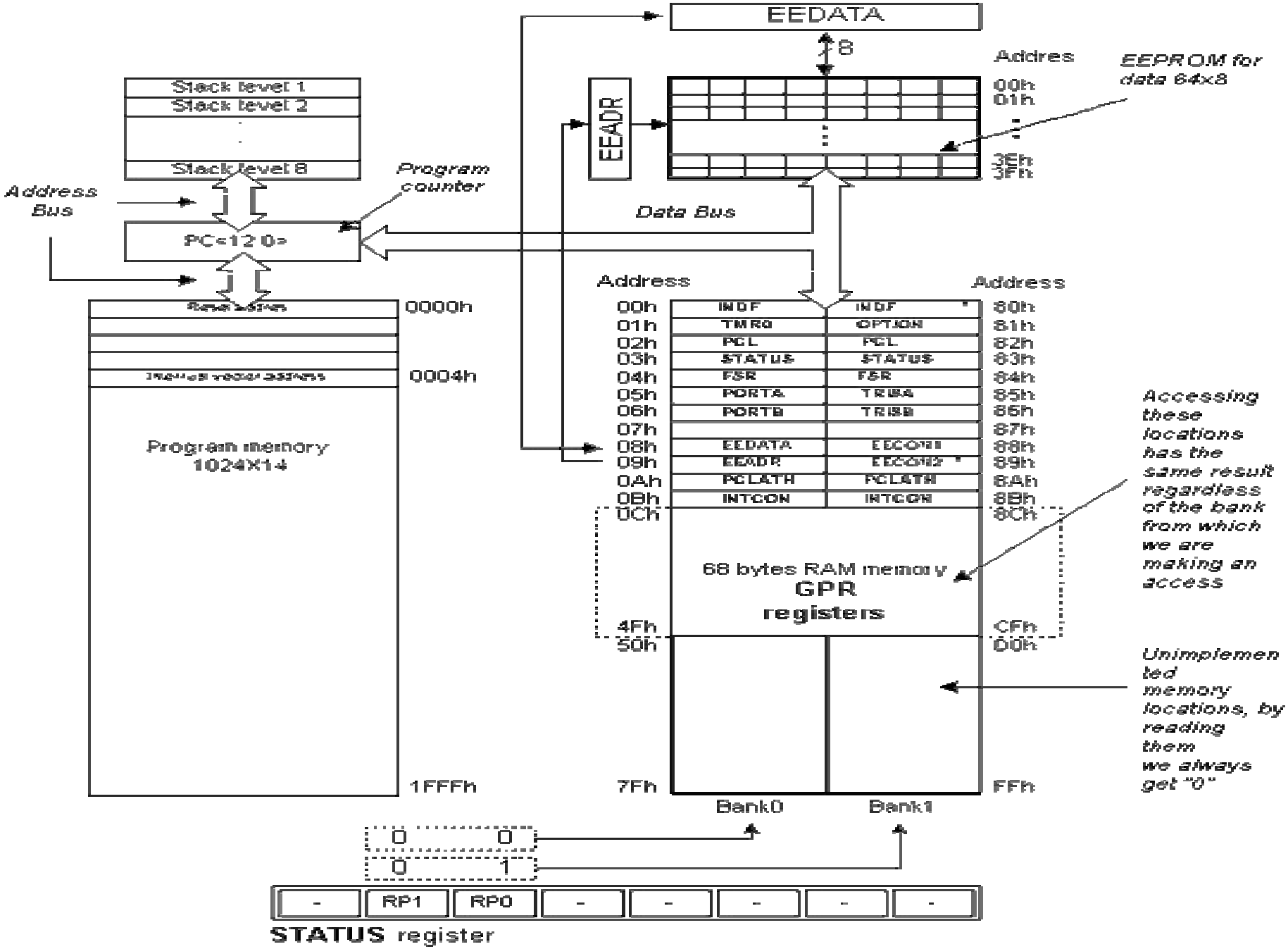
- PIC16F84 has two separate memory blocks, for data and for program.
- EEPROM memory with GPR and SFR registers in RAM memory make up the data block, while FLASH memory makes up the program block.

Program Memory

- Program memory has been carried out in FLASH technology.
- The size of program memory is 1024 locations with 14 bits width where locations zero and four are reserved for reset and interrupt vector.

Data Memory

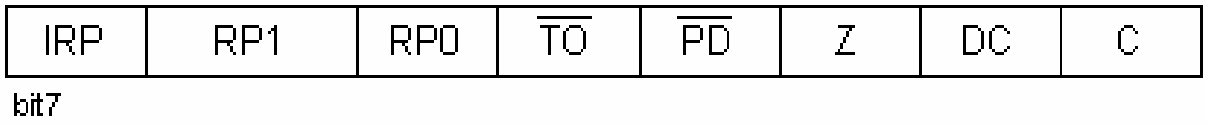
- Data memory consists of EEPROM and RAM memories.
- EEPROM memory consists of 64 eight bit locations.
- EEPROM is not directly addressable, but is accessed indirectly through EEADR and EEDATA registers.
- EEPROM memory usually serves for storing important parameters.
- RAM memory for data occupies space on a memory map from location 0x0C to 0x4F which comes to 68 locations (**GPR registers**).
- **SFR registers** take up first 12 locations in banks 0 and 1.



Memory organization of microcontroller PIC16F84

Memory Banks

- Memory map is divided in 'width' to two areas called 'banks'.
- Selecting one of the banks is done via RP0 bit in STATUS register.



Example:

bcf STATUS, RP0

Instruction BCF clears bit RP0 in STATUS register and thus sets up bank 0.

bsf STATUS, RP0

Instruction BSF sets the bit RP0 in STATUS register and thus sets up bank1.

What would happen if the wrong bank was selected?

BANK0 macro

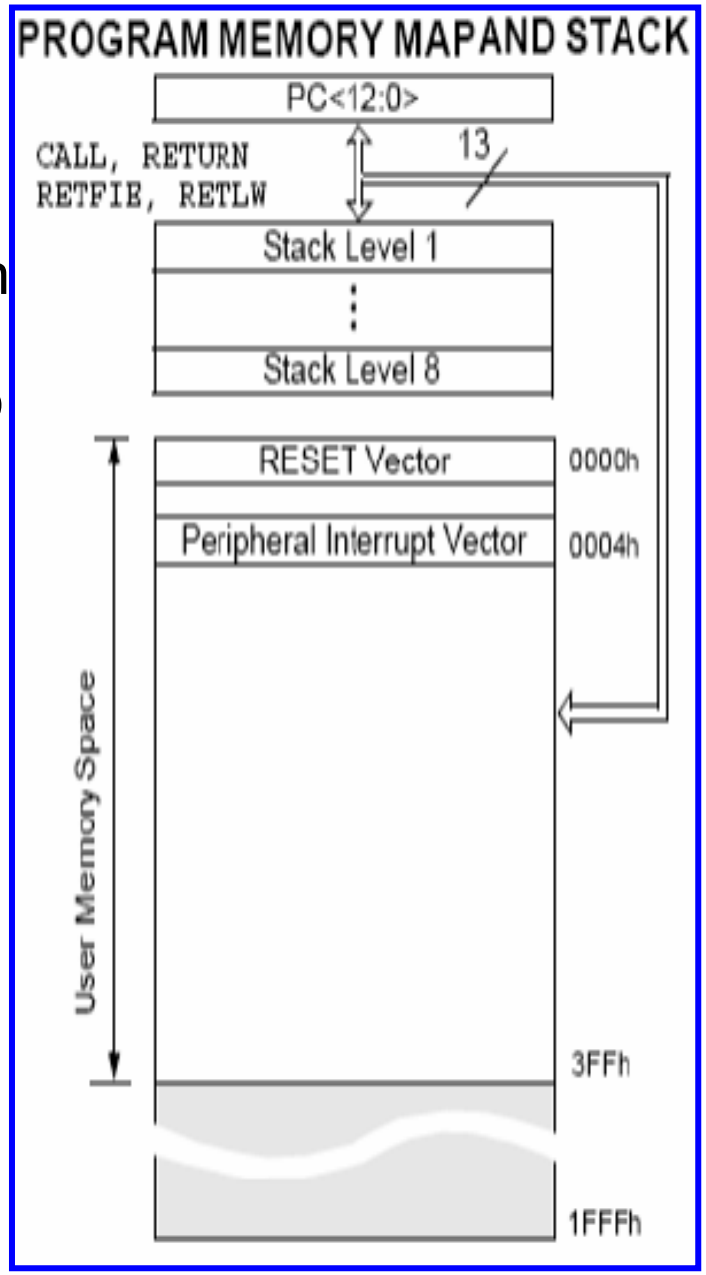
```
bcf STATUS, RP0   ;Select memory bank 0
endm
```

BANK1 macro

```
bsf STATUS, RP0   ;Select memory bank 1
endm
```

Stack:

- PIC16F84 has a 13-bit stack with 8 levels.
- Its basic role is to keep the value of PC after a jump from the main program to an address of a subprogram .
- In order for a program to know how to go back to the point where it started from, it has to return the value of a PC from a stack.
- When moving from a program to a subprogram, PC is being pushed onto a stack. When executing instructions such as RETURN, RETLW or RETFIE which were executed at the end of a subprogram, PC was taken from a stack so that program could continue where was stopped before it was interrupted. These operations of placing on and taking off from a program counter stack are called PUSH and POP.



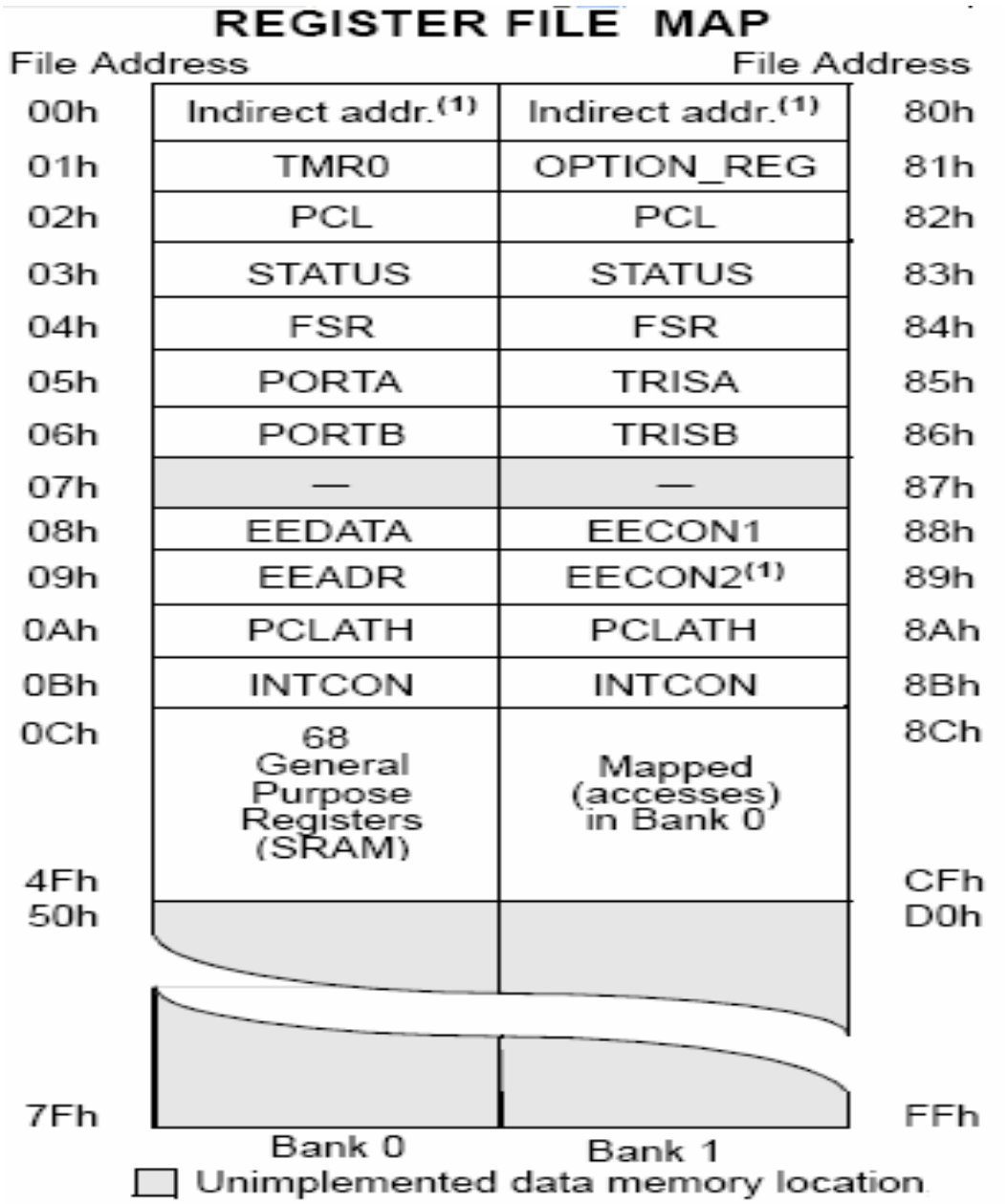
Data Memory:

Data memory is partitioned into two areas:

- SFR Area.
- GPR Area.

The GPR area allow greater than 116 bytes of GP RAM.

The data memory can be accessed either directly using the absolute address of each register file or indirectly through the file select register (FSR).



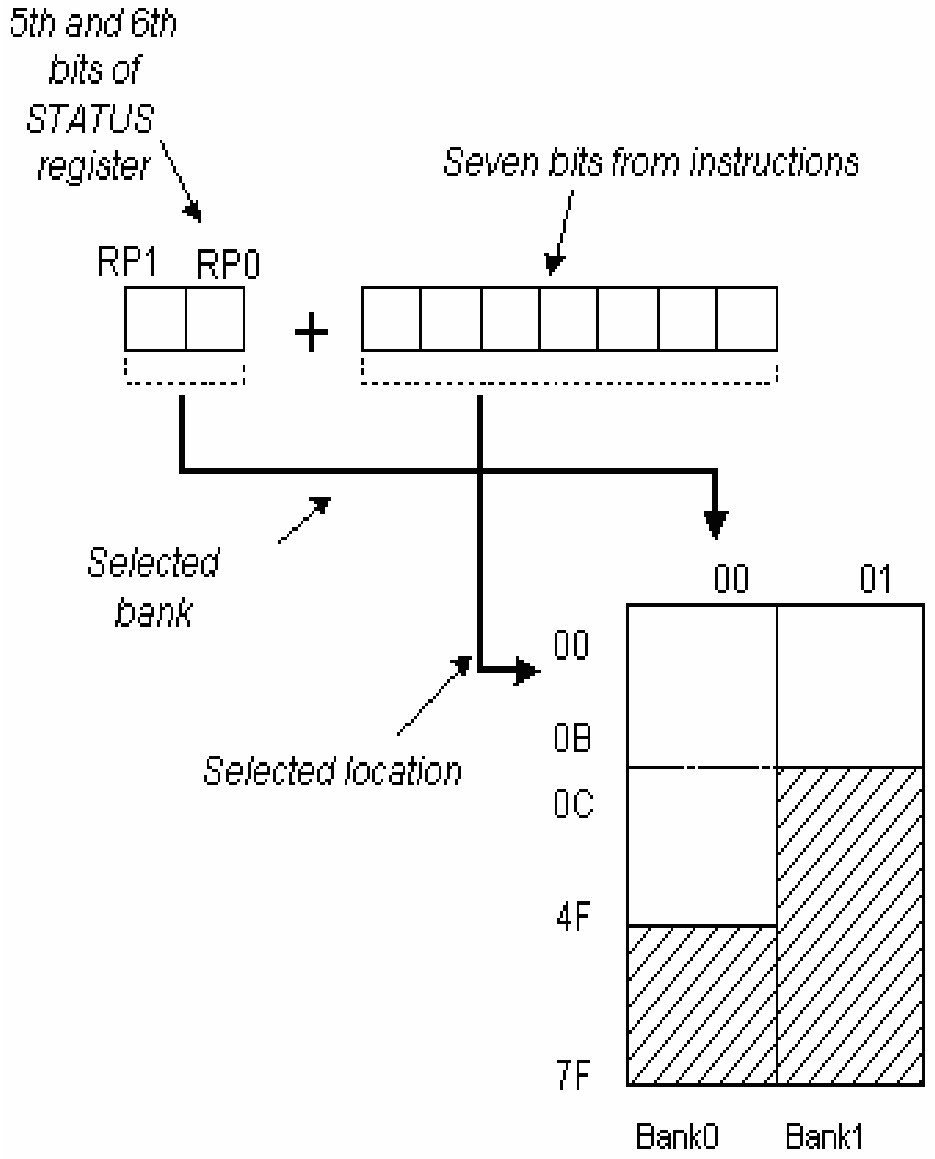
Addressing Modes:

- RAM memory locations can be accessed directly or indirectly.

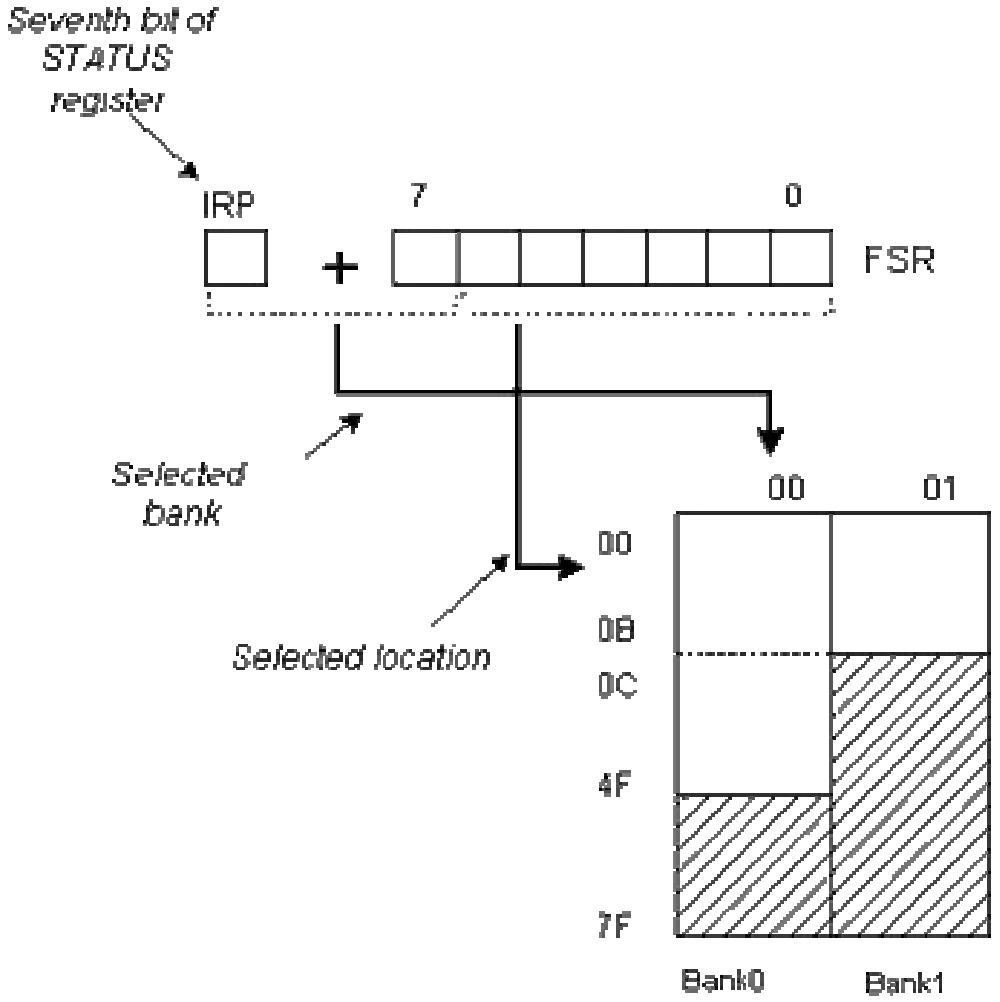
Direct Addressing:

- Direct Addressing is done through a 9-bit address.
- Example:

```
Bsf STATUS, RP0      ;Bankl
movlw 0xFF           ;w=0xFF
movwf TRISA          ;address of
TRISA register is taken from
instruction movwf
```



- **Indirect Addressing:**
- Indirect unlike direct addressing does not take an address from an instruction but derives it from IRP bit of STATUS and FSR registers.
- Addressed location is accessed via INDF register which in fact holds the address indicated by a FSR.

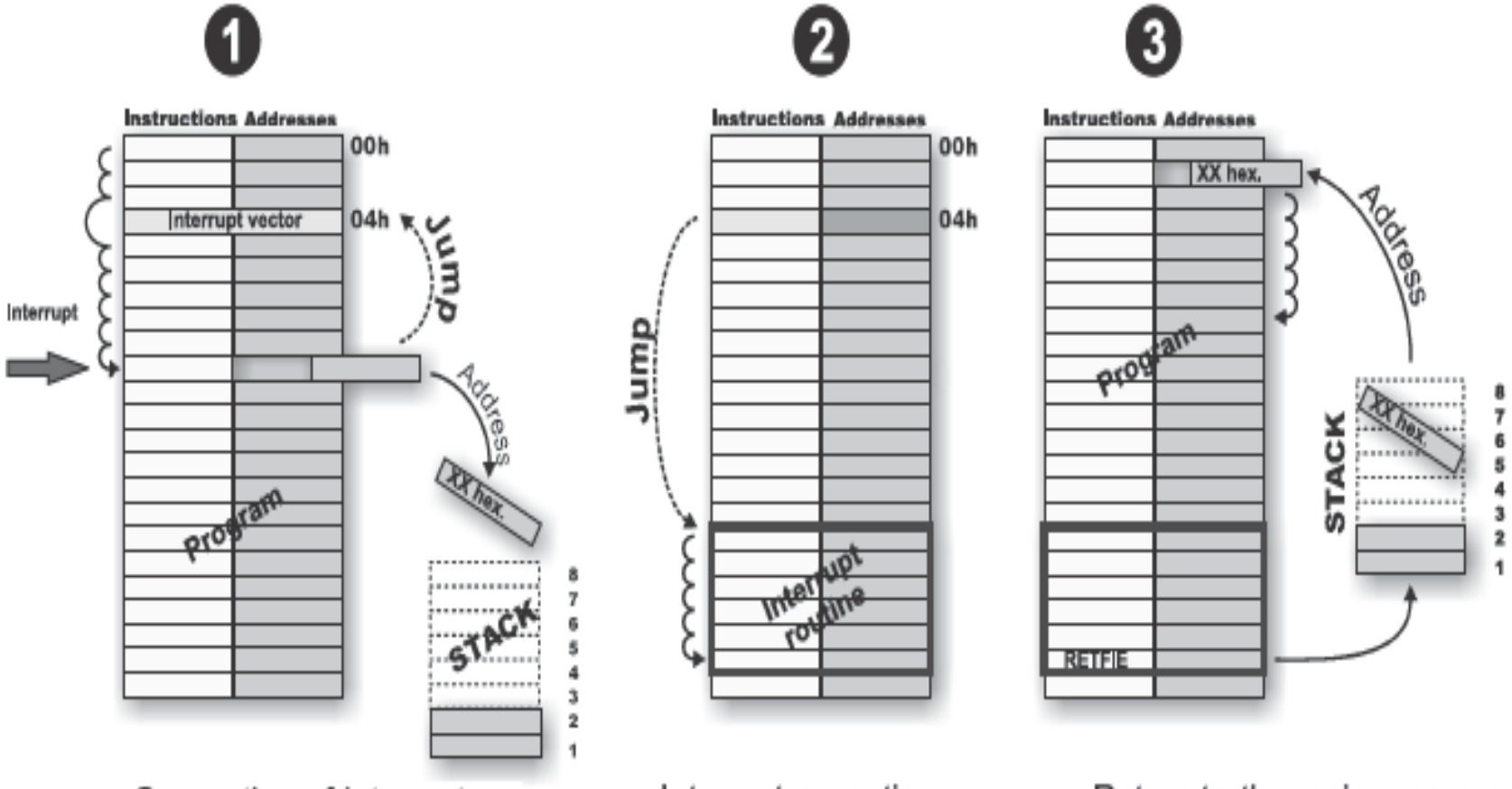


Indirect addressing

Interrupts:

Interrupts are a mechanism which enables MC to respond to some events, regardless of what MC is doing at that time.

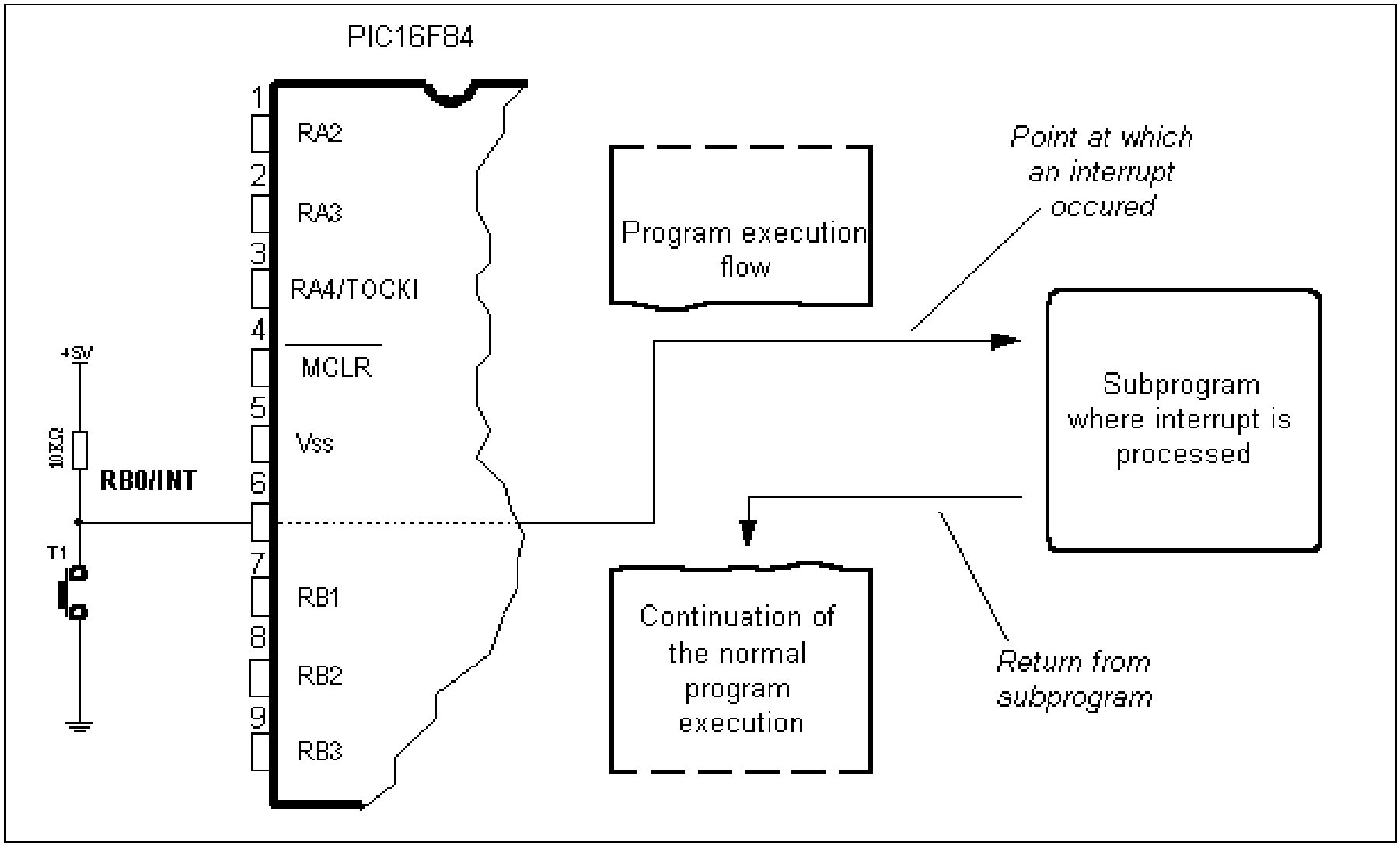
Each interrupt changes the program flow, interrupts it and after executing an interrupt routine, it continues from that same point on.



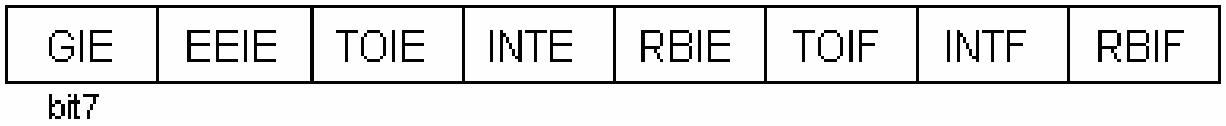
Generation of interrupt

Interrupt execution

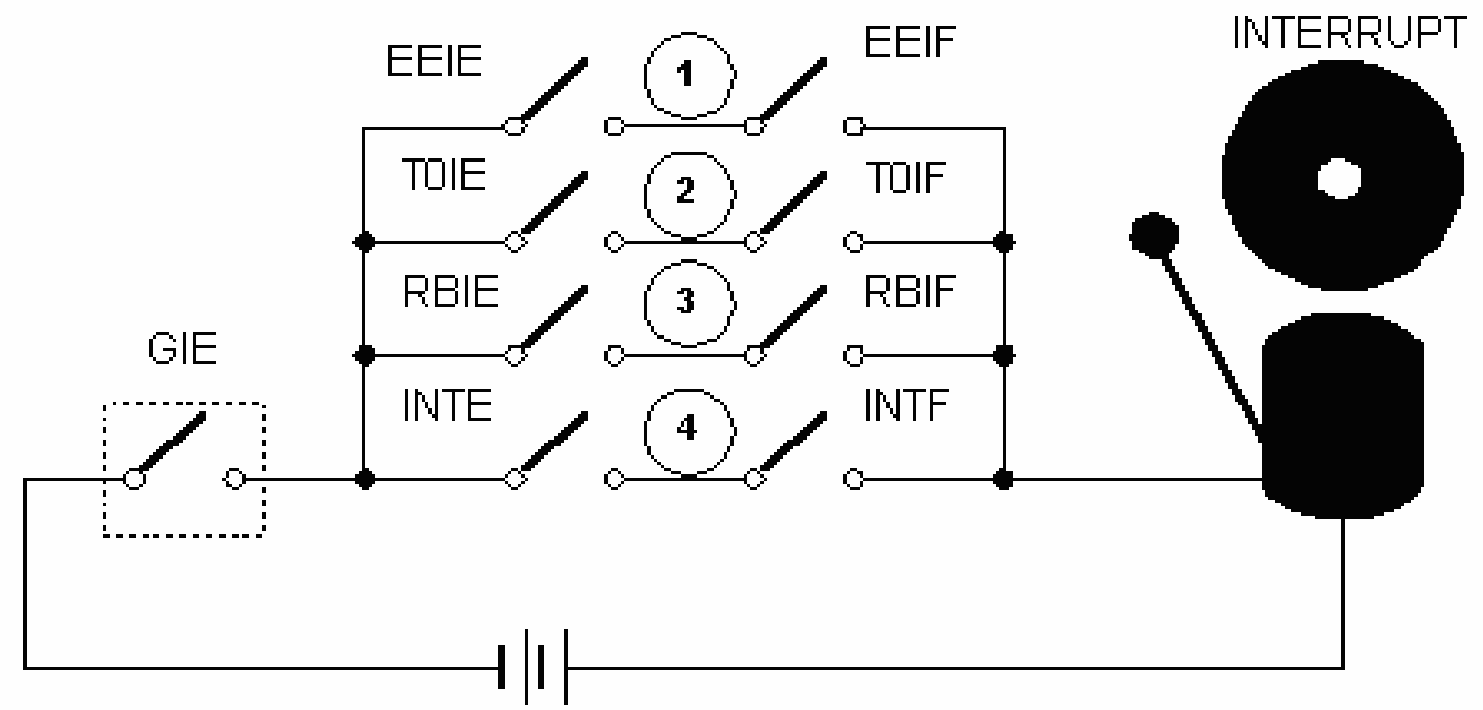
Return to the main program


PIC16F84
1
RA2
2
RA3
3
RA4/TOCKI
4
MCLR
5
Vss
6
RB0/INT
7
RB1
8
RB2
9
RB3
+5V
10KΩ
T1
Program execution flow
Point at which an interrupt occured
Subprogram where interrupt is processed
Continuation of the normal program execution
Return from subprogram

INTCON Register:



Bit 7: **GIE** (*Global Interrupt Enable bit*): enables or disables all INTs.
Bit 6: **EEIE** (*EEPROM Write Complete Interrupt Enable bit*): enables an INT at the end of a writing routine to EEPROM.
If EEIE and EEIF are set simultaneously , an INT will occur.
bit 5: **T0IE** (*TMR0 Overflow Interrupt Enable bit*): enables INTs during TMR0 overflow.
If T0IE and T0IF are set simultaneously, interrupt will occur.
Bit 4: **INTE** (*INT External Interrupt Enable bit*): enables external INT from pin RB0/INT.
Bit 3 **RBIE** (*RB port change Interrupt Enable bit*): enables INTs to occur at the change of status of pins 4, 5, 6, and 7 of port B.
If RBIE and RBIF are simultaneously set, an INT will occur.
Bit 2: **T0IF** (*TMR0 Overflow Interrupt Flag bit*): Overflow of counter TMR0.
Bit must be cleared in program in order for an INT to be detected.
Bit 1: **INTF** (*INT External Interrupt Flag bit*) External INT occurred.
If a rising or falling edge was detected on pin RB0/INT, INTF is set.
Bit 0: **RBIF** (*RB Port Change Interrupt Flag bit*): informs about changes on pins 4, 5, 6 and 7 of port B.



Simplified outline of PIC16F84 microcontroller interrupt

PIC16F84 has four interrupt sources:

1. Termination of writing data to EEPROM
2. TMR0 interrupt caused by timer overflow
3. Interrupt during alteration on RB4, RB5, RB6 and RB7 pins of port B.
4. External interrupt from RB0/INT pin of microcontroller.

External interrupt on RB0/INT pin:

External interrupt on RB0/INT pin is triggered by rising signal edge (if bit INTEDG=1 in OPTION<6> register), or falling edge (if INTEDG=0). When correct signal appears on INT pin, INTF bit is set in INTCON register. INTF bit (INTCON<1>) must be cleared in interrupt routine, so that interrupt wouldn't occur again while going back to the main program. Interrupt can be turned off by resetting INTE control bit (INTCON<4>).

Interrupt during a TMR0 counter overflow:

Overflow of TMR0 counter (from FFh to 00h) will set T0IF (INTCON<2>) bit. This is very important interrupt because many real problems can be solved using this interrupt.

Interrupt upon a change on pins 4, 5, 6 and 7 of port B

•Change of input signal on PORTB <7:4> sets RBIF (INTCON<0>) bit.
•Four pins RB7, RB6, RB5 and RB4 of port B, can trigger an interrupt which occurs when status on them changes from logic one to logic zero, or vice versa.
•For pins to be sensitive to this change, they must be defined as input. If any one of them is defined as output, interrupt will not be generated at the change of status. If they are defined as input, their current state is compared to the old value which was stored at the last reading from port B.

Interrupt upon finishing write-subroutine to EEPROM

This interrupt is of practical nature only. Since writing to one EEPROM location takes about 10ms (which is a long time in the notion of a MC), it doesn't pay off to a MC to wait for writing to end.
Thus interrupt mechanism is added which allows the MC to continue executing the main program, while writing in EEPROM is being done in the background.
When writing is completed, interrupt informs the MC that writing has ended.
EEIF bit, through which this informing is done, is found in EECON1 register.
Occurrence of an interrupt can be disabled by resetting the EEIE bit.

Interrupt initialization:

Touse an interrupt mechanism of a MC, some initialization tasks need to be performed.
By initialization we define to what interrupts the MC will respond, and which ones it will ignore.

```
clrf  INTCON              ; all interrupts disabled
movlw B'00010000'         ; external interrupt only is enabled
bsf   INTCON, GIE         ; occurrence of interrupts allowed
```

The above example shows initialization of external interrupt on RB0 pin of a MC. Occurrence of other interrupts is not allowed, and interrupts are disabled altogether until GIE bit is set to one.