



# **Distributed & Embedded Real-Time Systems**

## **(0640751)**

**Lecture (5)**

### **DERTS Design Requirements (1): Microcontroller Architecture & Programming**

**Prof. Kasim M. Al-Aubidy**  
Philadelphia University

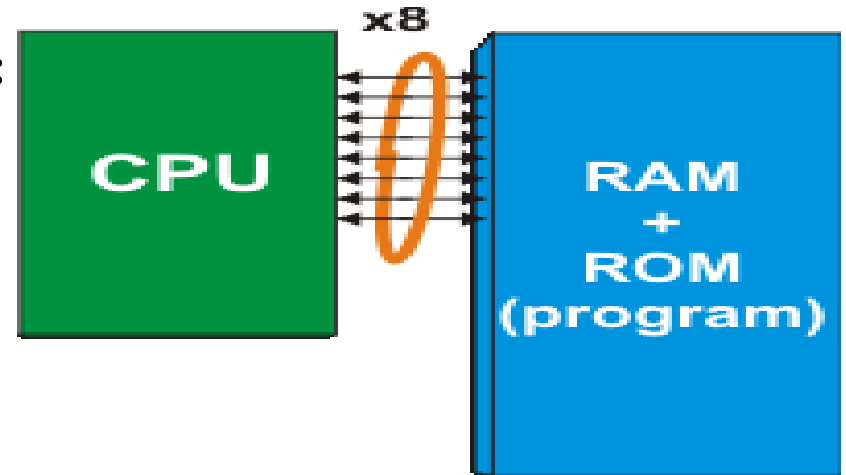
# Lecture Outline:

- Features of microcomputers and microcontrollers.
- Microcontroller Architecture.
- Microcontroller Instruction Set.
- Microcontroller Programming.
- Memory Organization.
- Addressing Modes.

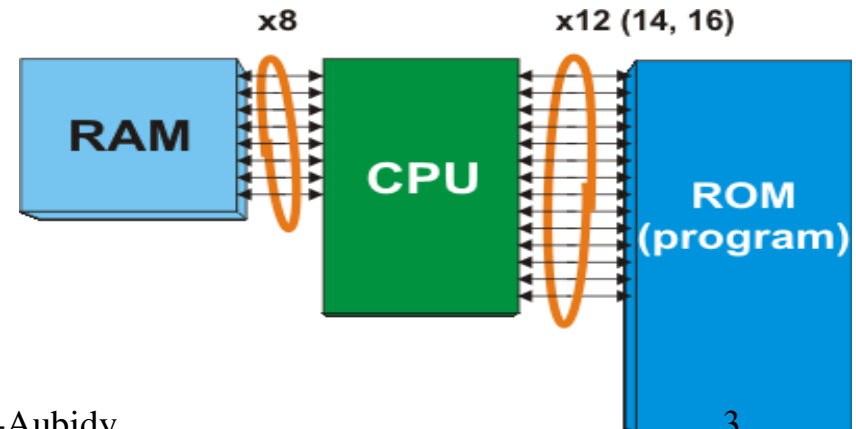
## INTERNAL ARCHITECTURE

- All MCs use one of two basic design models:  
*Harvard Architecture* and *von-Neumann architecture*.
- They represent two different ways of exchanging data between CPU and memory.

- **VON-NEUMANN ARCHITECTURE:**

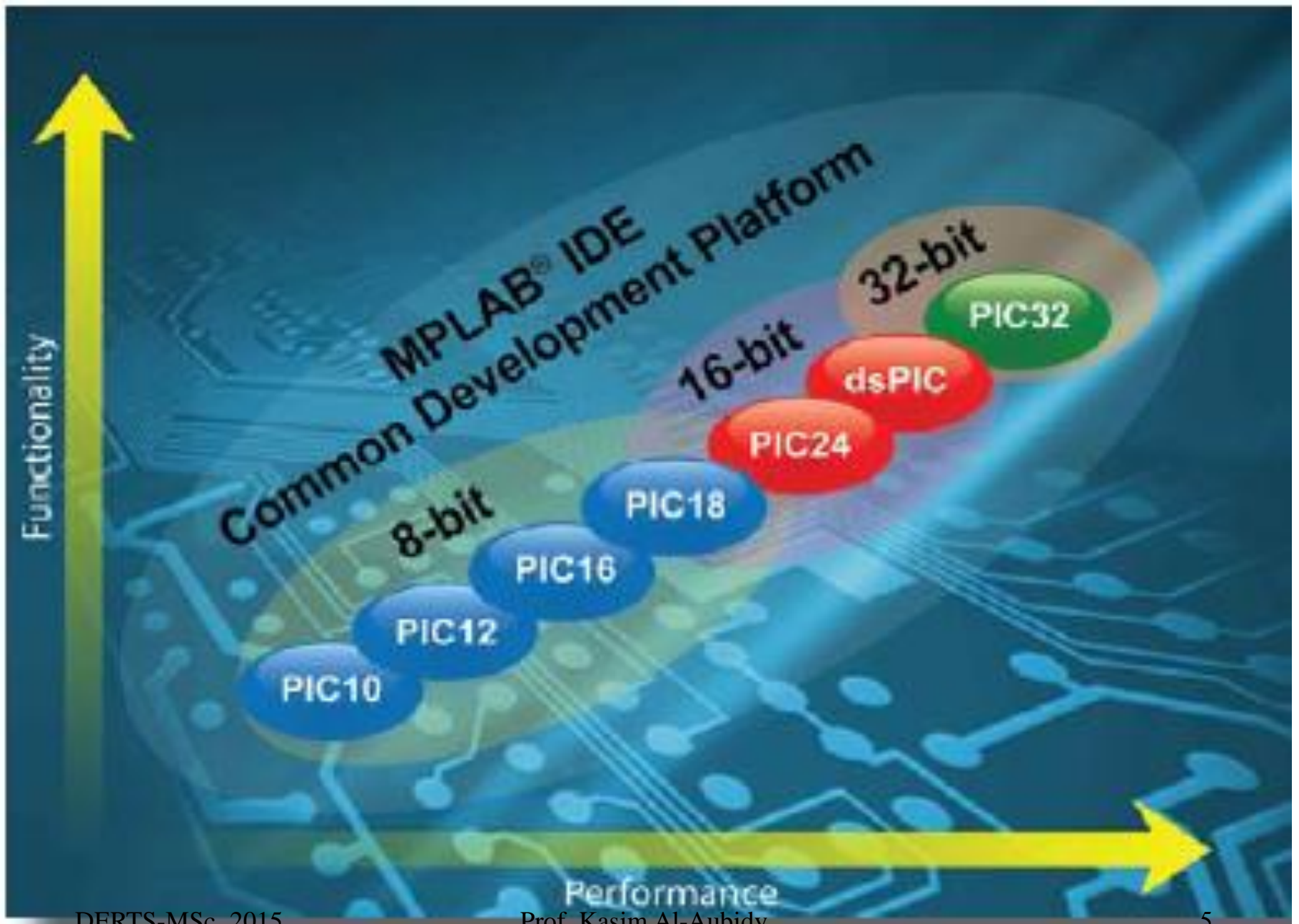


- **HARVARD ARCHITECTURE:**



# CISC and RISC

- MCs with Harvard architecture are called "RISC MCs". MCs with von-Neumann's architecture are called 'CISC microcontrollers'.
- The PIC16F84 MC has a RISC architecture.
- Harvard architecture is a newer concept than von-Neumann's.
- In Harvard architecture, data bus and address bus are separate. Thus a greater flow of data is possible through the CPU, and of course, a greater speed of work.
- PIC16F84 uses 14 bits for instructions which allows for all instructions to be one word instructions.
- It is also typical for Harvard architecture to have fewer instructions than von-Neumann's, and to have instructions usually executed in one cycle.
- The PIC16F84 MC has 35 instructions. All of these instructions are executed in one cycle except for jump and branch instructions.



# Popular PIC MCU Families

**PIC10:** Extremely small footprint, 6-pins

**PIC12:** Low-cost, easy-to-use, 8-pins

**PIC16:** NEW Enhanced Mid-Range core optimized for C with simplified memory map

**PIC18:** High 8-bit performance optimized for C with advanced communication peripherals, low-power, up to 128 KB Flash and 80-pins

**PIC24:** 16-bit families for more memory and faster peripherals including low power and high performance

**dsPIC® DSCs:** Digital signal control with motor control and power conversion peripherals, seamless migration with PIC24 MCUs

**PIC32:** Up to 80 MHz of 32-bit performance, compatible with 8- & 16-bit devices

- **Broad portfolio of more than 550 PIC microcontrollers**
  - From .5K to 512 KB Flash
  - From 0.5 to 80 MIPS performance
  - Multiple package options from 6- to 100-pins
  - nanoWatt XLP™ for eXtreme Low Power, <20 nA Sleep mode
- **Comprehensive technical documentation and free software**
  - Easy to get your designs done fast
  - Free software for USB, TCP-IP, ZigBee®, touch sensing, display and more
  - Leverage thousands of app notes, code examples and software libraries
- **MPLAB® IDE is absolutely free and the MPLAB tool suite supports ALL of Microchip's 8-, 16- and 32-bit microcontrollers**
  - Easy code migration
  - Free C Compiler without code size limitations
  - User-friendly, inexpensive programming and debug tools
  - Low-cost demo boards help speed up prototyping efforts
- **Easy-to-Use, Faster Time-to-Market**
  - C-code friendly with industry-leading code efficiency
  - PIC Architecture is easy to learn, easy to use
- **Easy migration with pin and code compatibility**
  - One MCU platform for all of your applications

- **Wide product availability and shortest lead times in the industry**
  - Worldwide fulfillment channels
  - Long product life cycles – we are still manufacturing the original PIC MCUs
- **The only supplier to bring USB, LCD, Ethernet, Touch Sensing and CAN to the 8-bit market**
  - Industry-leading integrated peripherals
  - Integrated nanoWatt XLP technology
  - Communication peripherals (SPI, I<sup>2</sup>C™, UART, USB, wireless)
  - Analog (8-, 10- and 12-bit ADC, comparators)
- **World-class, 24/7 technical support and training**
  - World-wide field application engineers
  - Built to support over 60,000 customers
  - Comprehensive web seminars, videos, hands-on training, “Lunch & Learns” and customer conferences
  - Leverage on-line community support from other developers on



Family	ROM [Kbytes]	RAM [bytes]	Pins	Clock Freq. [MHz]	A/D Inputs	Resolution of ADC	Comparators	8/16-bit Timers	Serial Comm.	PWM Outputs	Others
<b>Base-Line 8-bit architecture, 12-bit Instruction Word Length</b>											
PIC10FXXX	0.375-0.75	16 - 24	6 - 8	4 - 8	0 - 2	8	0 - 1	1 x 8	-	-	-
PIC12FXXX	0.75 - 1.5	25 - 38	8	4 - 8	0 - 3	8	0 - 1	1 x 8	-	-	EEPROM
PIC16FXXX	0.75 - 3	25 - 134	14 - 44	20	0 - 3	8	0 - 2	1 x 8	-	-	EEPROM
PIC16HVXXX	1.5	25	18 - 20	20	-	-	-	1 x 8	-	-	V <sub>dd</sub> = 15V
<b>Mid-Range 8-bit architecture, 14-bit Instruction Word Length</b>											
PIC12FXXX	1.75 - 3.5	64 - 128	8	20	0 - 4	10	1	1 - 2 x 8 1 x 16	-	0 - 1	EEPROM
PIC12HVXXX	1.75	64	8	20	0 - 4	10	1	1 - 2 x 8 1 x 16	-	0 - 1	-
PIC16FXXX	1.75 - 14	64 - 368	14 - 64	20	0 - 13	8 or 10	0 - 2	- 2 x 8 1 x 16	USART I2C SPI	0 - 3	-

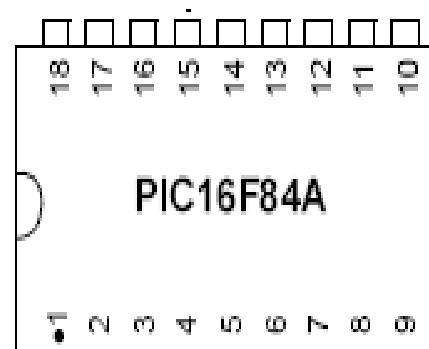
## 18-pin *Enhanced* FLASH/EEPROM 8-Bit Microcontroller

### High Performance RISC CPU Features:

- Only 35 single word instructions to learn
- All instructions single-cycle except for program branches which are two-cycle
- Operating speed: DC - 20 MHz clock input
- 1024 words of program memory
- 68 bytes of Data RAM
- 64 bytes of Data EEPROM
- 14-bit wide instruction words
- 8-bit wide data bytes
- 15 Special Function Hardware registers

### Peripheral Features:

- 13 I/O pins with individual direction control
- High current sink/source for direct LED drive
  - 25 mA sink max. per pin
  - 25 mA source max. per pin
- TMR0: 8-bit timer/counter with 8-bit programmable prescaler



## Special Microcontroller Features:

- 10,000 erase/write cycles *Enhanced* FLASH Program memory typical
- 10,000,000 typical erase/write cycles EEPROM Data memory typical
- EEPROM Data Retention > 40 years
- In-Circuit Serial Programming™ (ICSP™) - via two pins
- Power-on Reset (POR), Power-up Timer (PWRT), Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own On-Chip RC Oscillator for reliable operation
- Code protection
- Power saving SLEEP mode

## CMOS Enhanced FLASH/EEPROM Technology:

- Low power, high speed technology
- Fully static design
- Wide operating voltage range:
  - Commercial: 2.0V to 5.5V
  - Industrial: 2.0V to 5.5V
- Low power consumption:
  - < 2 mA typical @ 5V, 4 MHz
  - 15  $\mu$ A typical @ 2V, 32 kHz
  - < 0.5  $\mu$ A typical standby current @ 2V

# THE PIC16F887 BASIC FEATURES:

## **RISC architecture**

Only 35 instructions to learn

All single-cycle instructions except branches

## **Operating frequency 0-20 MHz**

## **Precision internal oscillator**

Factory calibrated

Software selectable frequency range of 8MHz to 31KHz

## **Power supply voltage 2.0-5.5V**

Consumption: 220uA (2.0V, 4MHz), 11uA (2.0 V, 32 KHz)  
50nA (stand-by mode)

## **Power-Saving Sleep Mode**

## **35 input/output pins**

High current source/sink for direct LED drive

software and individually programmable *pull-up* resistor

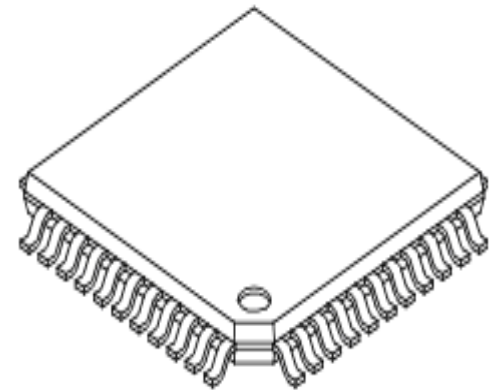
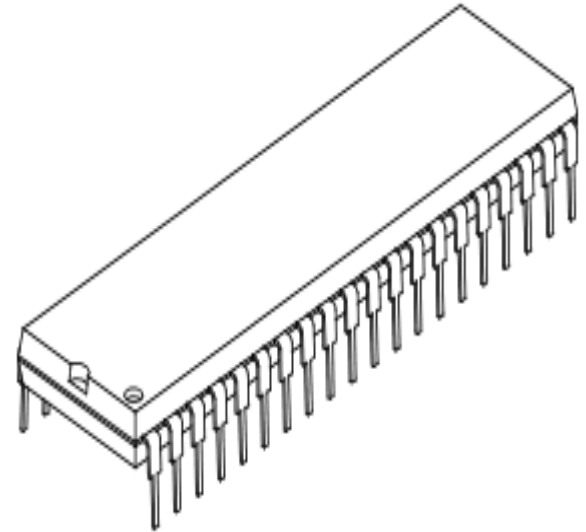
Interrupt-on-Change pin

## **8K ROM memory in FLASH technology**

Chip can be reprogrammed up to 100.000 times

## ***In-Circuit Serial Programming Option***

Chip can be programmed even embedded in the target device



# THE PIC16F887 BASIC FEATURES:

**256 bytes EEPROM memory**

Data can be written more than 1.000.000 times

**368 bytes RAM memory**

**A/D converter:**

14-channels

10-bit resolution

**3 independent timers/counters**

**Watch-dog timer**

**Analogue comparator module with**

Two analogue comparators

Fixed voltage reference (0.6V)

Programmable on-chip voltage reference

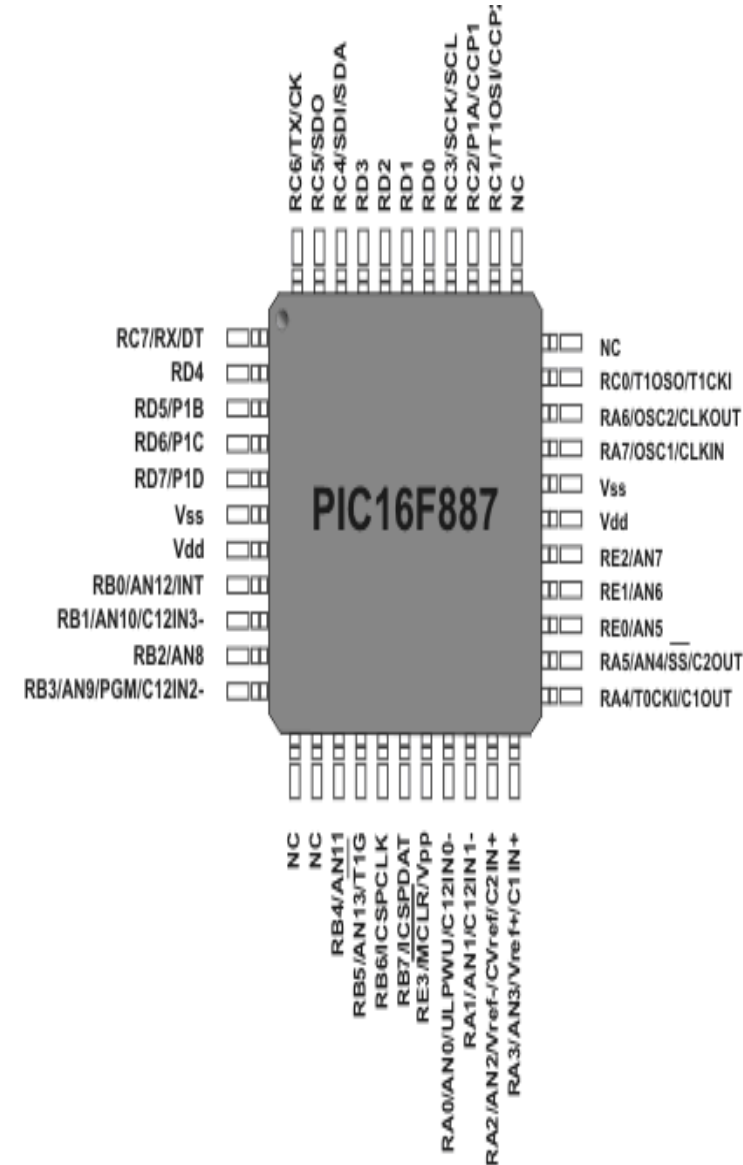
**PWM output steering control**

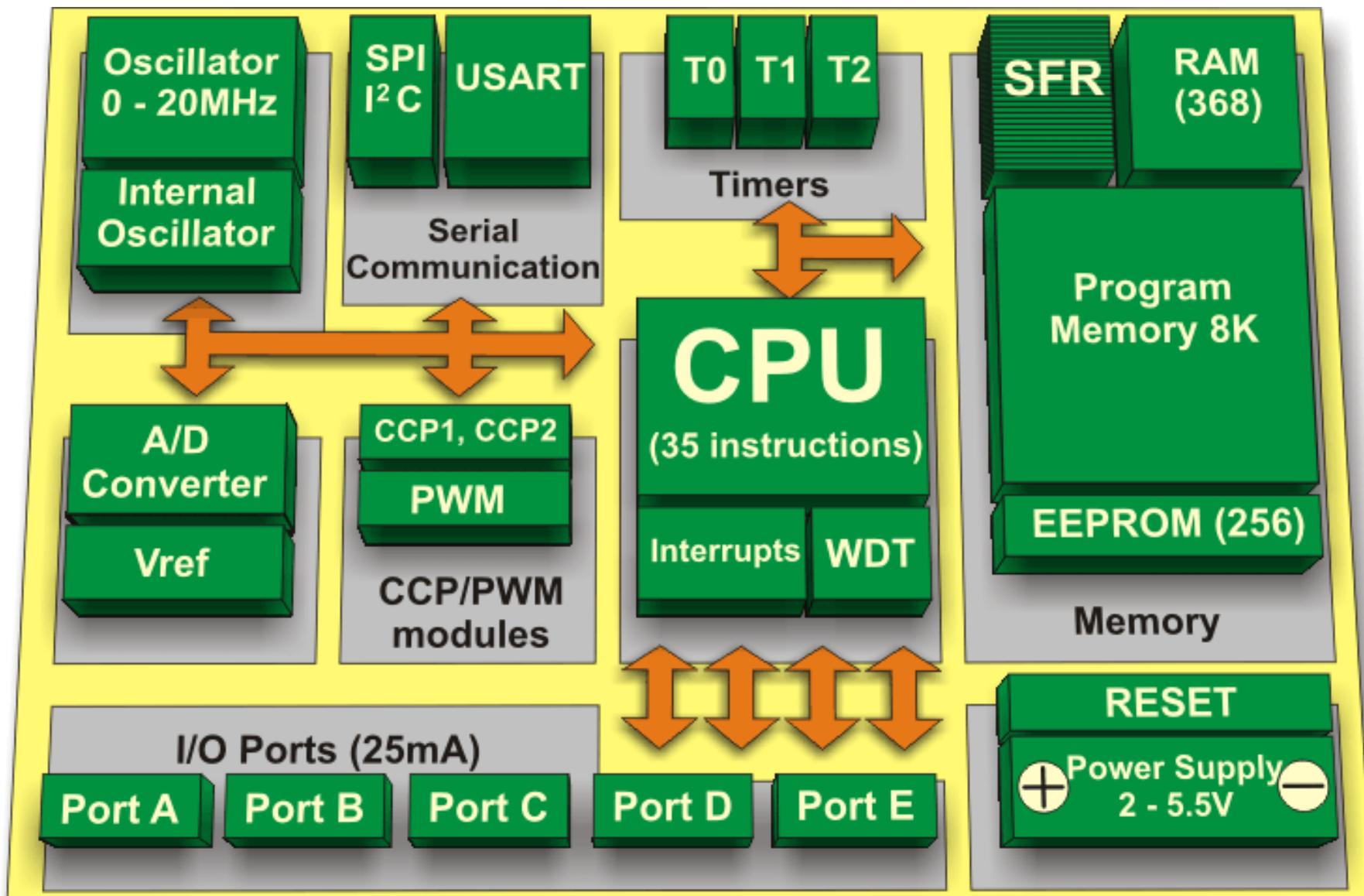
**Enhanced USART module**

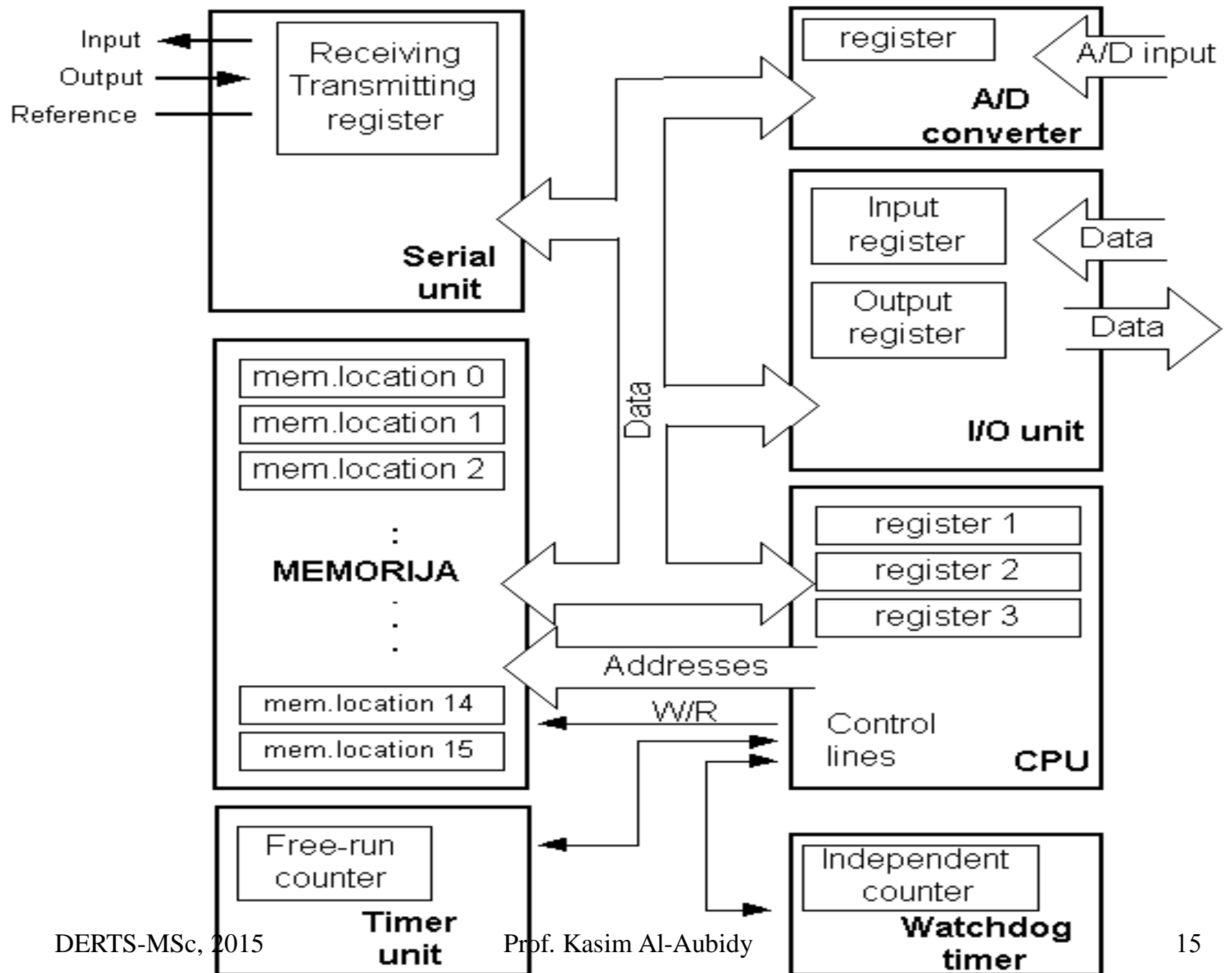
Supports RS-485, RS-232 and LIN2.0

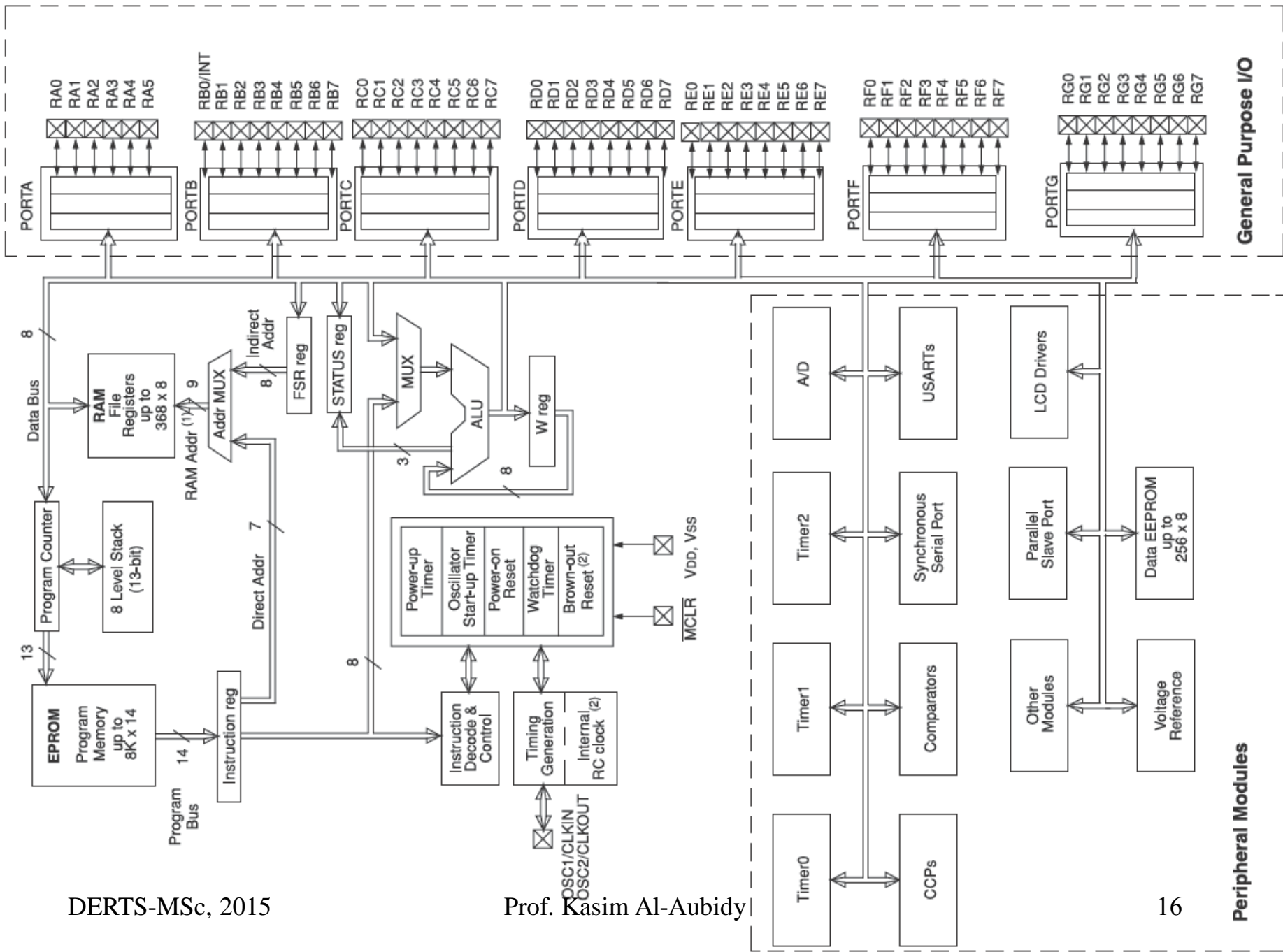
Auto-Baud Detect

**Master Synchronous Serial Port (MSSP)**

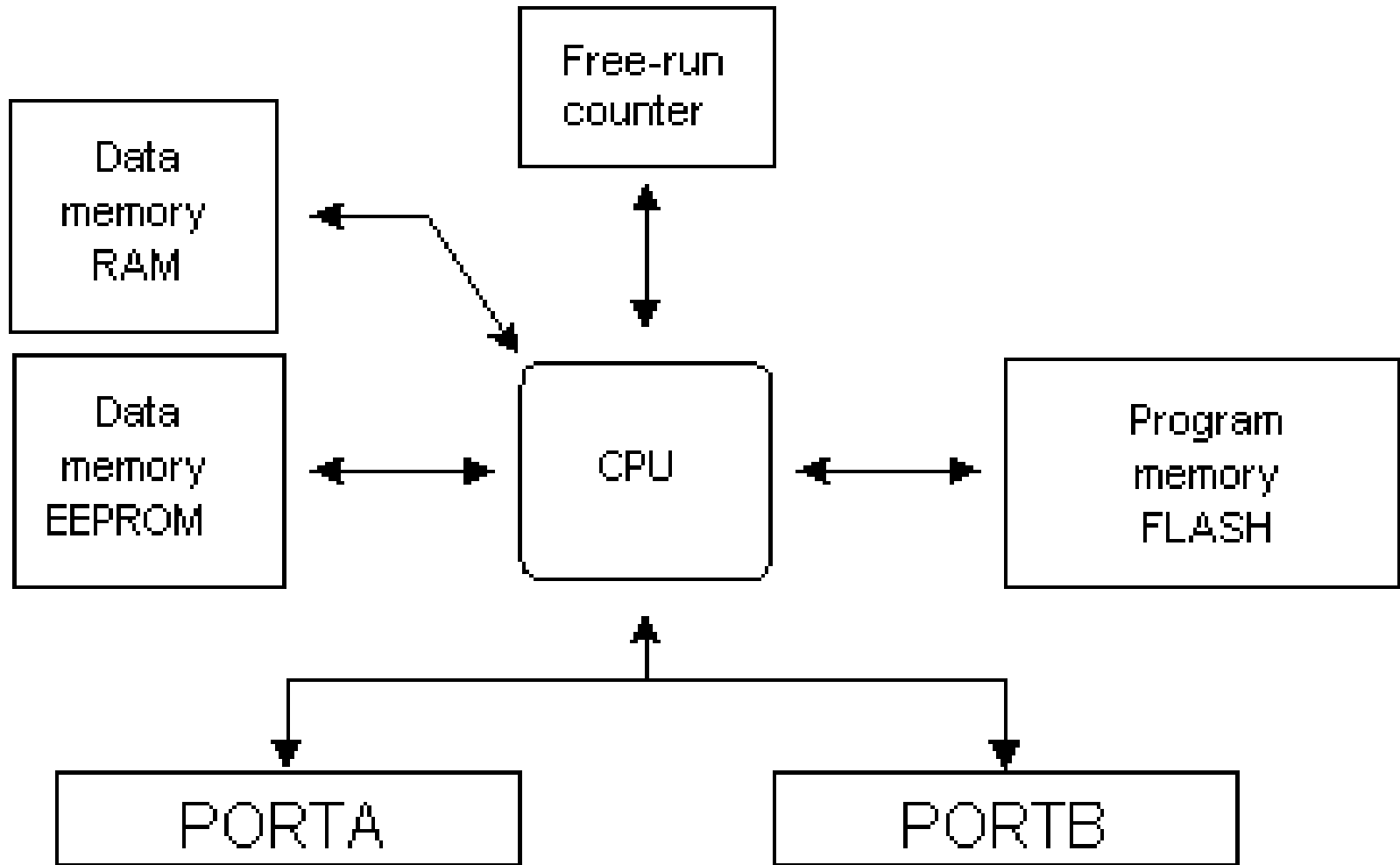




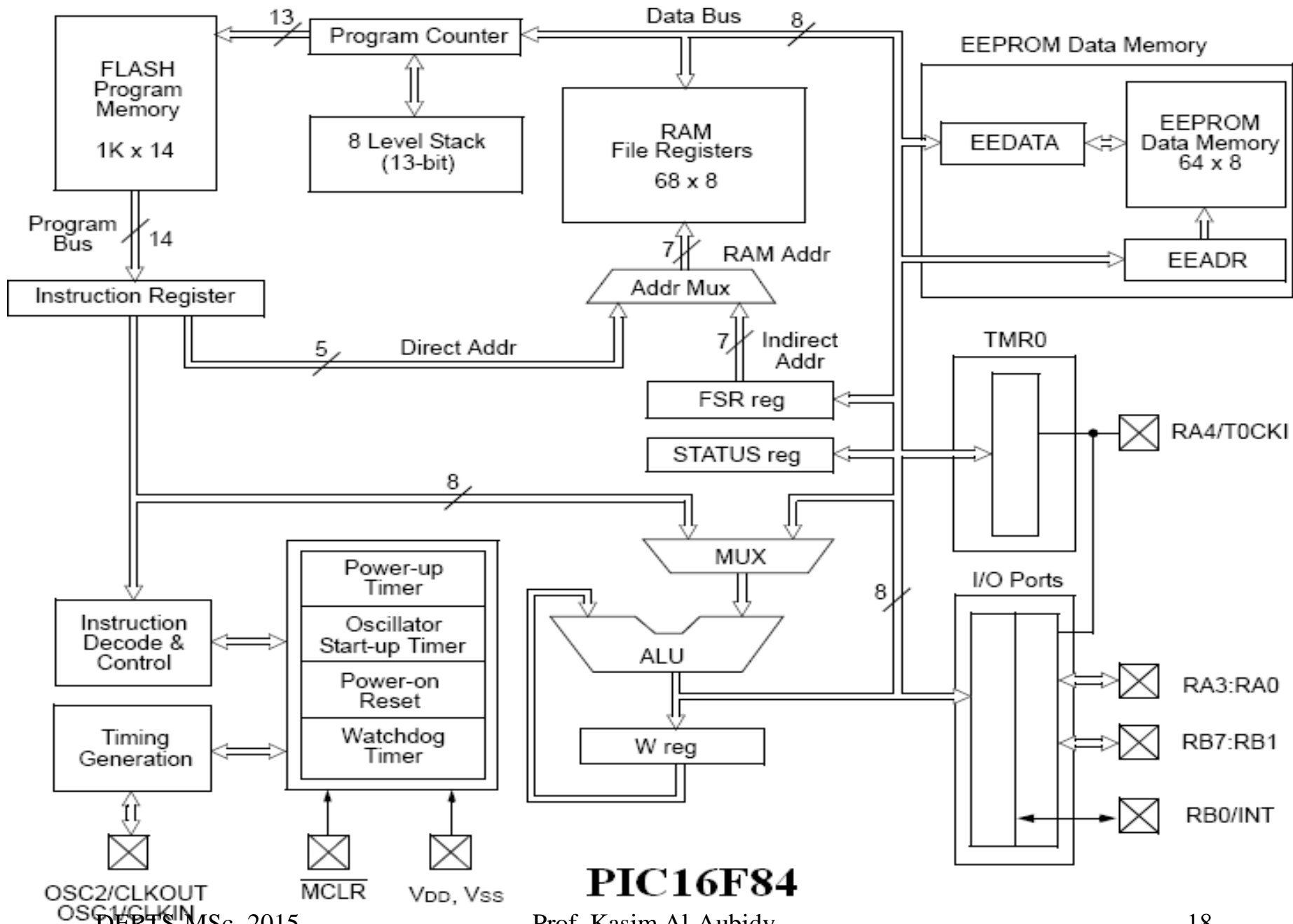








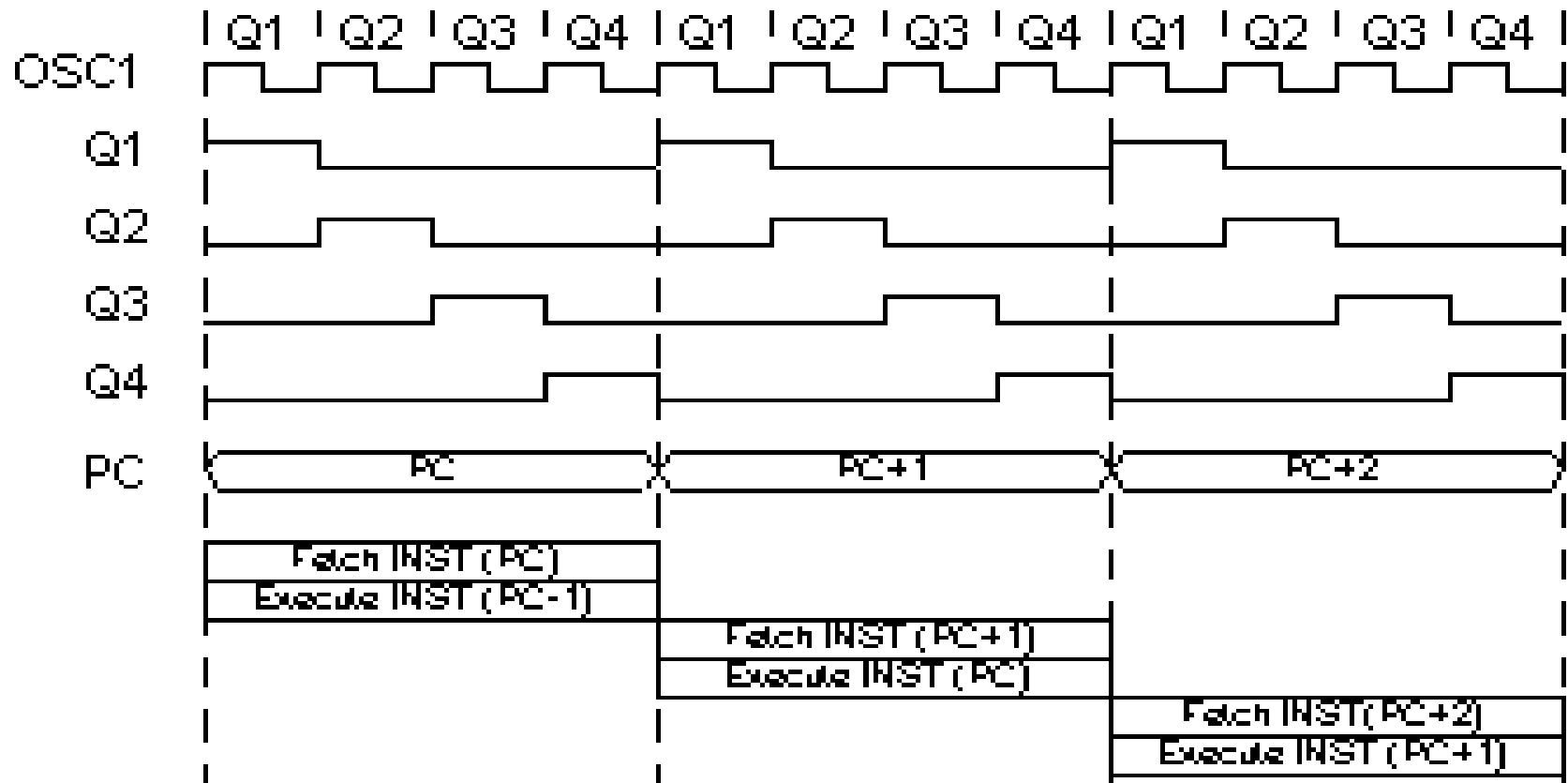
## PIC16F84 microcontroller outline



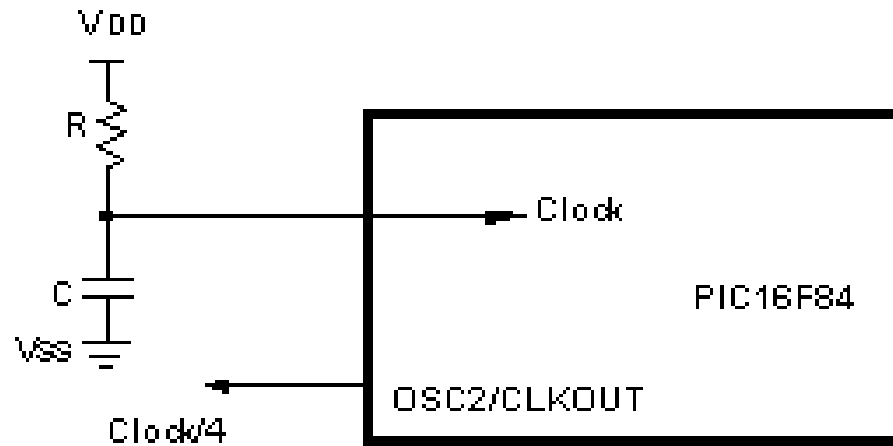
# PIC16F84

Prof. Kasim Al-Aubidy

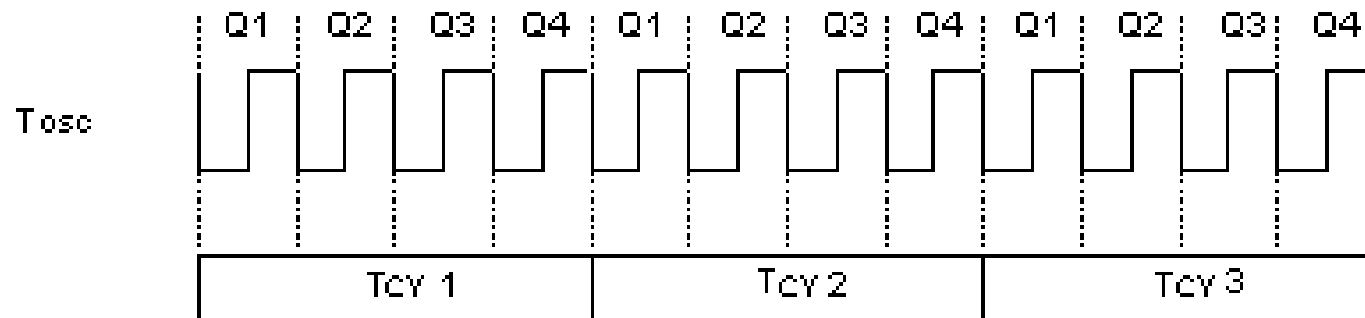
# Pipelining:



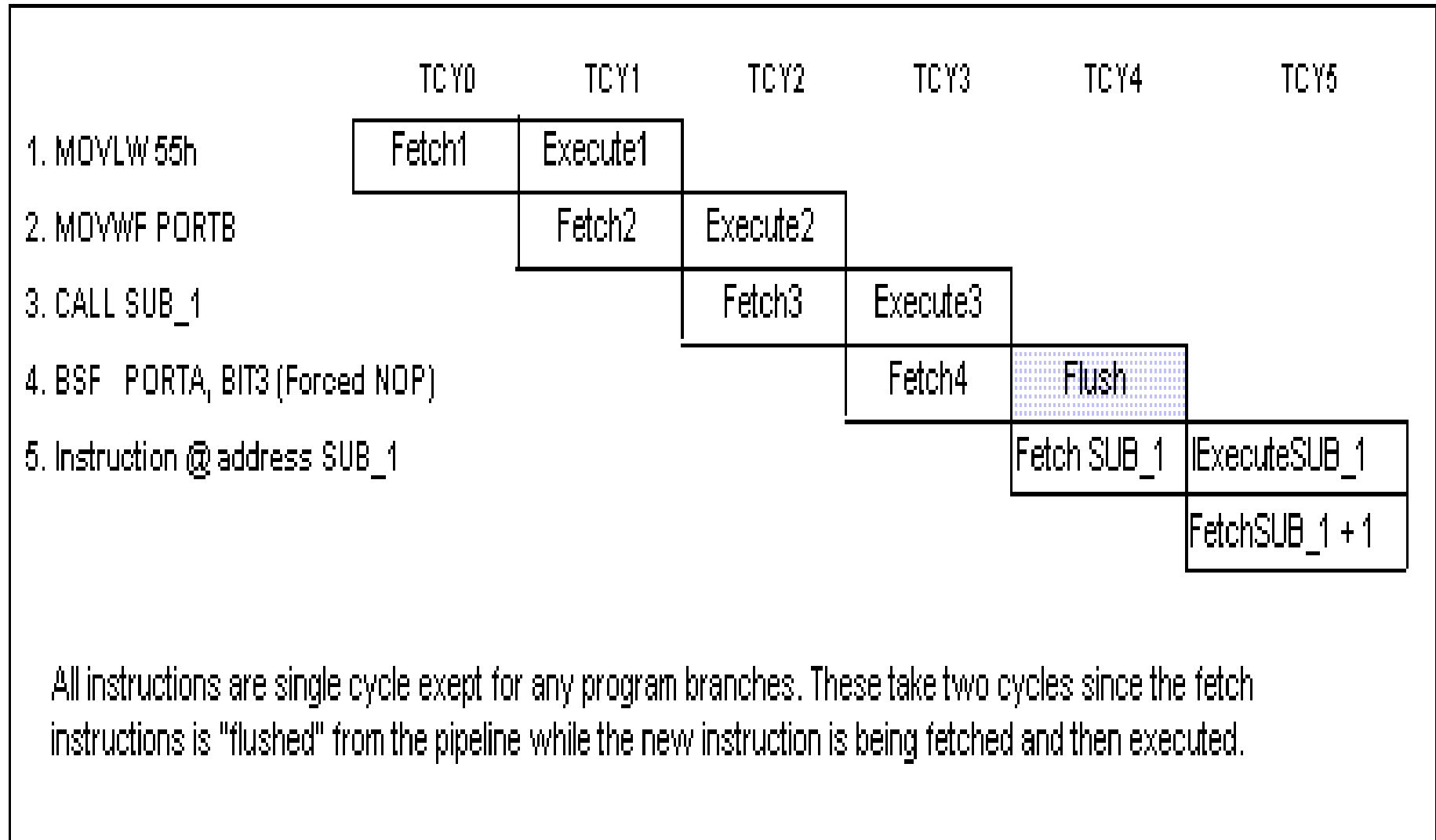
Clock/Instruction Cycle



Note: This pin can be configured as input/output pin

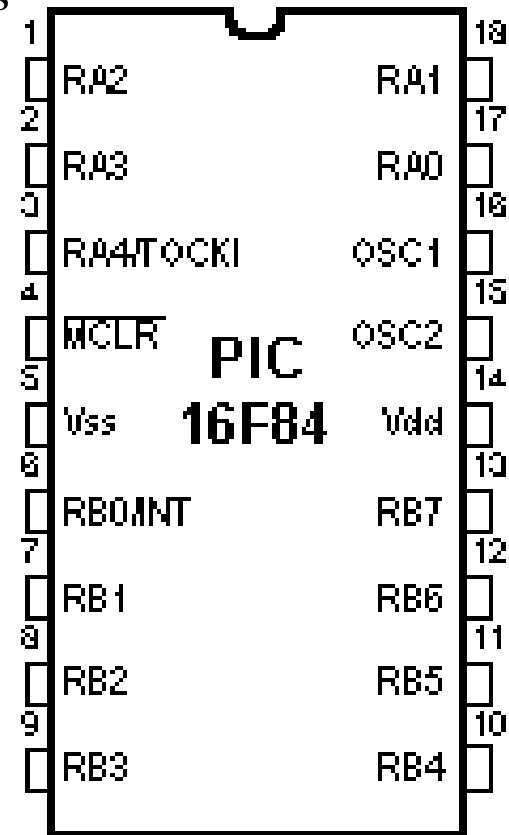


Relationship between a clock and a number of instruction cycles



### Instruction Pipeline Flow

- Pin no.1: **RA2** Second pin on port A.
- Pin no.2: **RA3** Third pin on port A.
- Pin no.3: **RA4** Fourth pin on port A. TOCK1 which functions as a timer is also found on this pin.
- Pin no.4: **MCLR** Reset i/p and Vpp programming voltage.
- Pin no.5: **Vss** Ground of power supply.
- Pin no.6: **RB0** Zero pin on port B. Interrupt input.
- Pin no.7: **RB1** First pin on port B.
- Pin no.8: **RB2** Second pin on port B.
- Pin no.9: **RB3** Third pin on port B.
- Pin no.10: **RB4** Fourth pin on port B.
- Pin no.11: **RB5** Fifth pin on port B.
- Pin no.12: **RB6** Sixth pin on port B. 'Clock' line in program mode.
- Pin no.13: **RB7** Seventh pin on port B. 'Data' line in program mode.
- Pin no.14: **Vdd** Positive power supply pole.
- Pin no.15: **OSC2** Pin for connecting with an oscillator.
- Pin no.16: **OSC1** Pin for connecting with an oscillator.
- Pin no.17: **RA2** Second pin on port A.
- Pin no.18: **RA1** First pin on port A.



# Instruction Set in PIC16Cxx MC Family

Complete set: 35 instructions.

MC Architecture: RISC microcontroller.

## Instruction Types:

### 1. Data Processing Operations:

- Copy data between registers.
- Manipulate data in a single register.
- Arithmetic operations.
- Logic operations.

### 2. Program Sequence Control Operations:

- Unconditional Jump.
- Conditional Jump.
- Call.

– Control.

## Word list

**f** any memory location in a microcontroller

**W** work register

**b** bit position in 'f' register

**d** destination bit

*label* group of eight characters which marks the beginning of a part of the program

**TOS** top of stack

[ ] option

< > bit position inside register

Example:

008C	movwf 0C
------	----------



# Data transfer

Transfer of data in a MC is done between W register and an 'f' register.

Mnemonic	Description	Operation	Flag	Cycle	Notes	
<b>Data transfer</b>						
MOVLW	k	Move constant to W	$k \rightarrow W$		1	
MOVWF	f	Move W to f	$W \rightarrow f$		1	
MOVF	f, d	Move f	$f \rightarrow d$	Z	1	1,2
CLRWF	-	Clear W	$0 \rightarrow W$	Z	1	
CLRF	f	Clear f	$0 \rightarrow f$	Z	1	2
SWAPF	f, d	Swap nibbles in f	$f(7:4), (3:0) \rightarrow f(3:0), (7:4)$		1	1,2

These instructions provide for:



- a constant being written in W register (MOVLW)
- data to be copied from W register onto RAM.
- data from RAM to be copied onto W register (or on the same RAM location, at which point only the status of Z flag changes).
- Instruction CLRF writes constant 0 in 'f' register,
- Instruction CLRWF writes constant 0 in register W.
- SWAPF instruction exchanges places of the 4-bit nibbles field inside a register.

# Arithmetic and logic

PIC MCs like most MCs supports only subtraction and addition.

Flags C, DC and Z are set depending on a result of addition or subtraction.

Logic unit performs AND, OR, EX-OR, complement (COMF) and rotation (RLF & RRF).

Arithmetic and logic						
ADDLW	k	Add constant and W	$W+1 \rightarrow W$	C,DC,Z	1	
ADDWF	f, d	Add W and f	$W+f \rightarrow d$	C,DC,Z	1	1,2
SUBLW	k	Subtract W from constant	$W-k \rightarrow W$	C,DC,Z	1	
SUBWF	f, d	Subtract W from f	$W-f \rightarrow d$	C,DC,Z	1	1,2
ANDLW	k	AND constant with W	$W.AND.k \rightarrow W$	Z	1	
ANDWF	f, d	AND W with f	$W.AND.f \rightarrow d$	Z	1	1,2
IORLW	k	OR constant with W	$W.OR.k \rightarrow W$	Z	1	
IORWF	f, d	OR W with f	$W.OR.f \rightarrow d$	Z	1	1,2
XORLW	k	Exclusive OR constant with W	$W.XOR.k \rightarrow W$	Z	1	1,2
XORWF	f, d	Exclusive OR W with f	$W.XOR.f \rightarrow d$	Z	1	
INCF	f, d	Increment f	$f+1 \rightarrow f$	Z	1	1,2
DECF	f, d	Decrement f	$f-1 \rightarrow f$	Z	1	1,2
RLF	f, d	Rotate Left f trough carry		C	1	1,2
RRF	f, d	Rotate Right f trough carry		C	1	1,2
COMF	f, d	Complement f	$f \rightarrow d$	Z	1	1,2

# Bit operations

Instructions BCF and BSF do setting or cleaning of one bit anywhere in the memory.

The CPU first reads the whole byte, changes one bit in it and then writes in the entire byte at the same place.

## Bit operations

BCF	f, b	Bit Clear f	$0 \rightarrow f(b)$		1	1,2
BSF	f, b	Bit Set f	$1 \rightarrow f(b)$		1	1,2

# Directing a program flow

- Instructions **GOTO**, **CALL** and **RETURN** are executed the same way as on all other microcontrollers, only stack is independent of internal RAM and limited to eight levels.
- '**RETLW k**' instruction is identical with **RETURN** instruction, except that before coming back from a subprogram a constant defined by instruction operand is written in W register.

## Directing a program flow

BTFSC	f, b	Bit Test f, Skip if Clear	jump if f(b)=0		1 (2)	3
BTFSS	f, b	Bit Test f, Skip if Set	jump if f(b)=1		1 (2)	3
DECFSZ	f, d	Decrement f, Skip if 0	f-1 → d, jump if Z=1		1(2)	1,2,3
INCFSZ	f, d	Increment f, Skip if 0	f+1 → d, jump if Z=0		1(2)	1,2,3
GOTO	k	Go to address	W.AND.k → W		2	
CALL	k	Call subroutine	W.AND.f → d		2	
RETURN	-	Return from Subroutine	W.OR.k → W		2	
RETLW	k	Return with constant in W	W.OR.f → d		2	
RETFIE	-	Return from interrupt	W.XOR.k → W		2	

# Look-up tables Design:

- This instruction 'RETLW k' enables us to design easily the Look-up tables (lists).
- We use them by determining data position on our table adding it to the address at which the table begins, and then we read data from that location.
- Table can be formed as a subprogram which consists of a series of 'RETLW k' instructions, where 'k' constants are members of the table.
- We write the position of a member of our table in W register, and using CALL instruction to call a subprogram which creates the table.
- The instruction **ADDWF PCL, f** adds the position of a W register member to the starting address of our table, found in PCL register, and so we get the real data address in program memory.
- When returning from a subprogram we will have in W register the contents of an addressed table member.

```
Main    movl 2
        call Lookup
Lookup  addwf PCL, f
        retlw k
        retlw k1
        retlw k2
        :
        :
        retlw kn
```

## Other instructions

NOP	-	No Operation			1	
CLRWDT	-	Clear Watchdog Timer	$0 \rightarrow \overline{\text{WDT}}, 1 \rightarrow \overline{\text{TO}}, 1 \rightarrow \overline{\text{PD}}$	$\overline{\text{TO}}, \overline{\text{PD}}$	1	
SLEEP	-	Go into standby mode	$0 \rightarrow \overline{\text{WDT}}, 1 \rightarrow \overline{\text{TO}}, 0 \rightarrow \overline{\text{PD}}$	$\overline{\text{TO}}, \overline{\text{PD}}$	1	

# Configuration Bits

- The configuration bits can be programmed (read as '0'), or left unprogrammed (read as '1'), to select various device configurations.
- These bits are mapped in program memory location 2007h.
- Address 2007h is beyond the user program memory space and it belongs to the special test/configuration memory space (2000h - 3FFFh). This space can only be accessed during programming.

## PIC16F84A CONFIGURATION WORD

R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u	R/P-u
CP	CP	CP	CP	CP	CP	CP	CP	CP	CP	$\overline{\text{PWRTE}}$	WDTE	FOSC1	FOSC0

bit13

bit 13-4 **CP**: Code Protection bit  
 1 = Code protection disabled  
 0 = All program memory is code protected

bit 3  $\overline{\text{PWRTE}}$ : Power-up Timer Enable bit  
 1 = Power-up Timer is disabled  
 0 = Power-up Timer is enabled

bit0

bit 2 **WDTE**: Watchdog Timer Enable bit  
 1 = WDT enabled  
 0 = WDT disabled

bit 1-0 **FOSC1:FOSC0**: Oscillator Selection bits  
 11 = RC oscillator  
 10 = HS oscillator  
 01 = XT oscillator  
 00 = LP oscillator

## AVAILABLE MEMORY IN PIC

# Memory Organization:

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

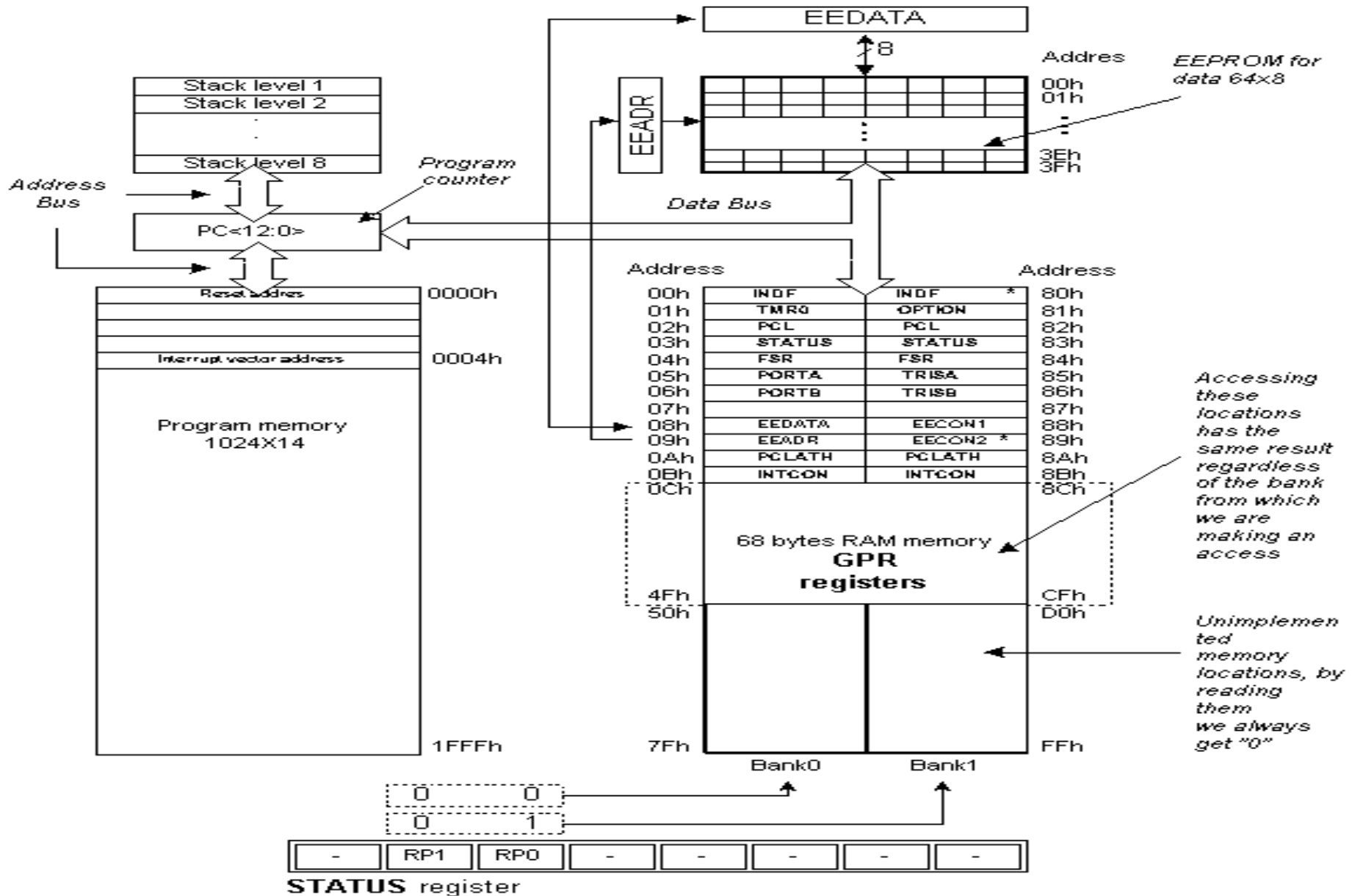
- PIC16F84 has two separate memory blocks, for data and for program.
- EEPROM memory with GPR and SFR registers in RAM memory make up the data block, while FLASH memory makes up the program block.

### 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.



bit7

**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 bank 1.

What would happen if the wrong bank was selected?

## BANK0 macro

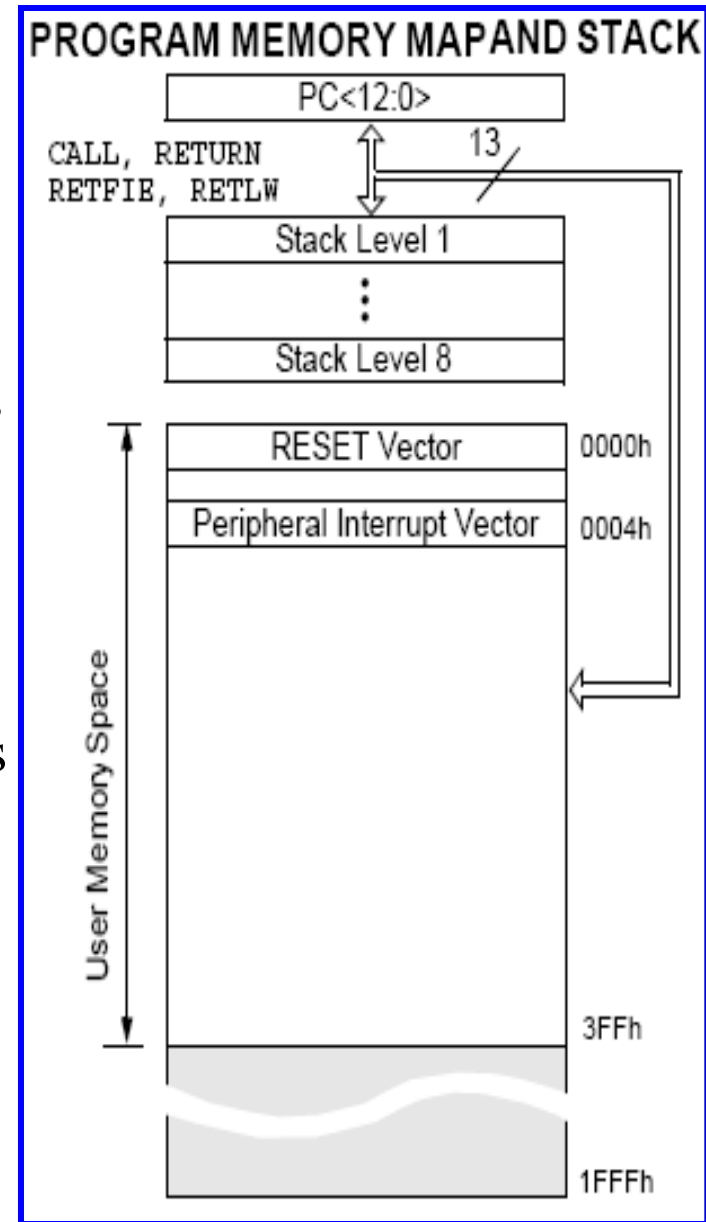
```
bcf STATUS, RP0 ;Select memory bank 0  
endm
```

## 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.



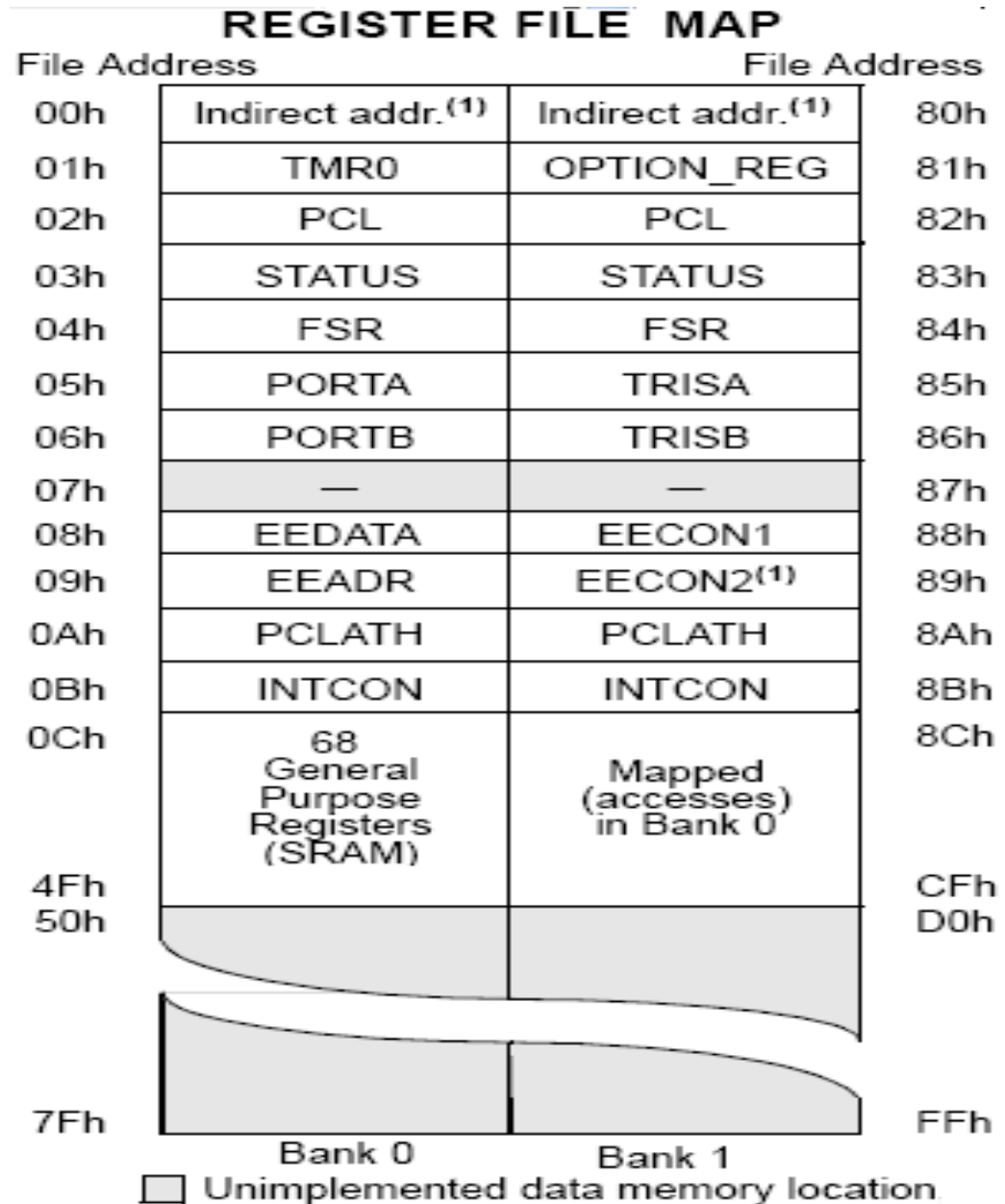
## Data Memory:

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:

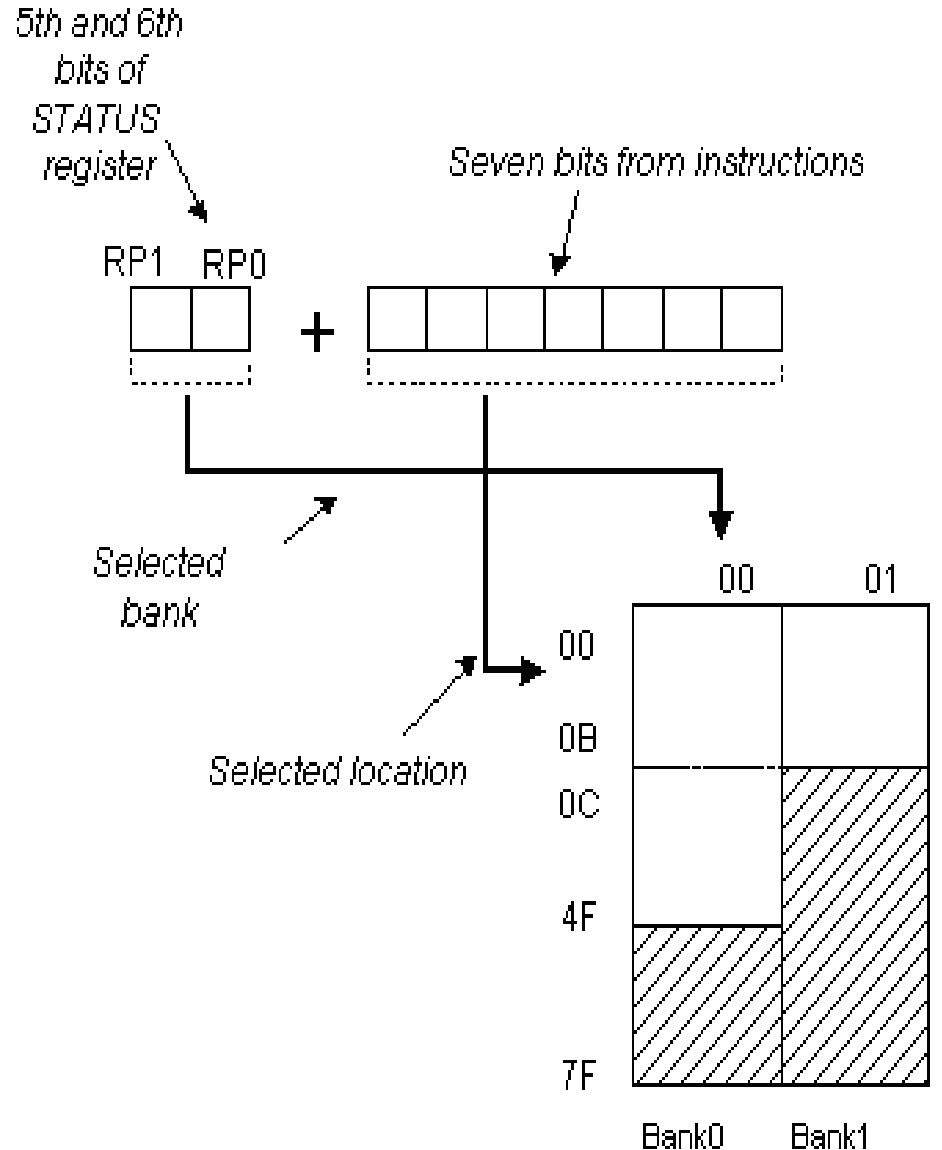
- RAM memory locations can be accessed directly or indirectly.

## Direct Addressing:

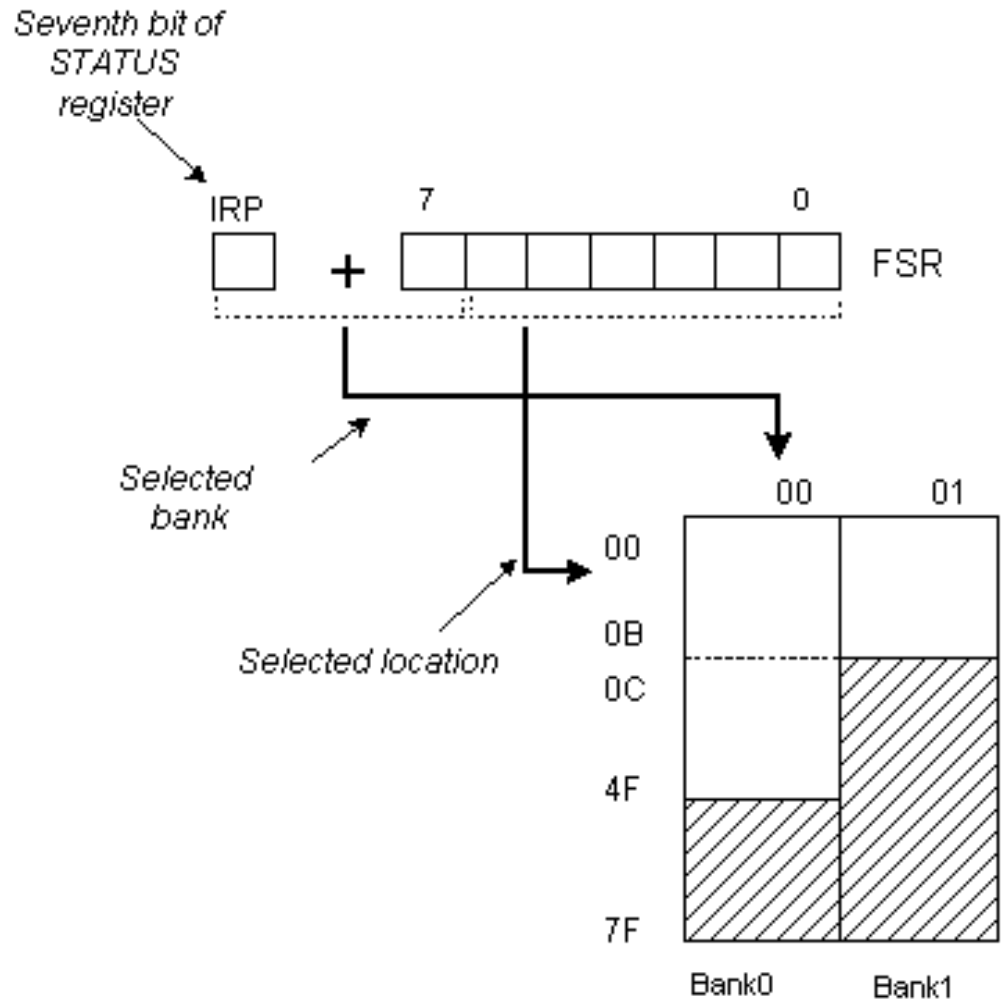
- Direct Addressing is done through a 9-bit address.
- Example:

```

Bsf STATUS, RP0      ;Bank1
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