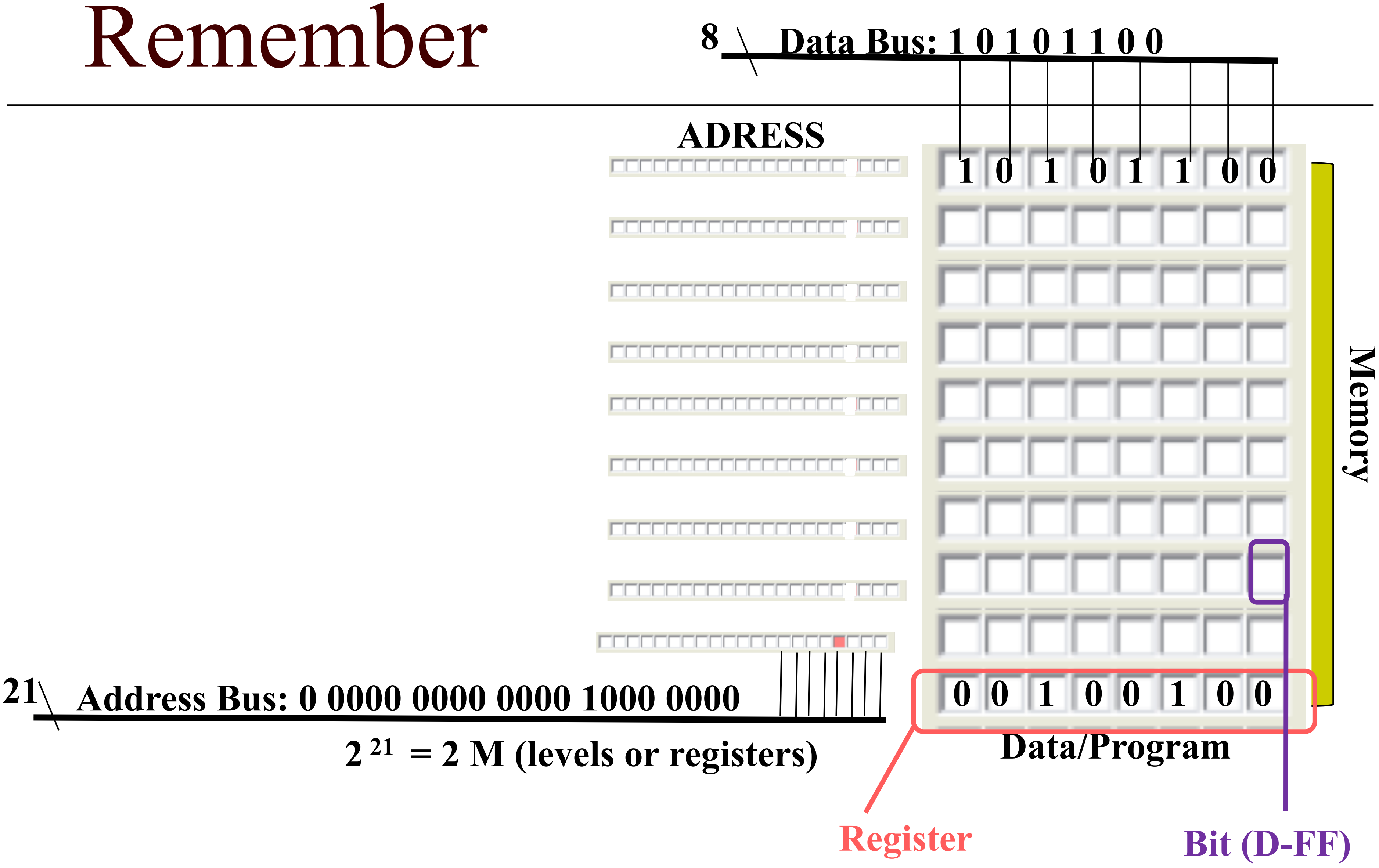
Chapter 2

Microcontroller Architecture—PIC18F Family

Updated 2/5/2019

Remember



8 Data Bus: 1 0 1 0 1 1 0 0

ADRESS

Memory

21 Address Bus: 0 0000 0000 0000 1000 0000

2^{21} = 2 M (levels or registers)

Data/Program

Register

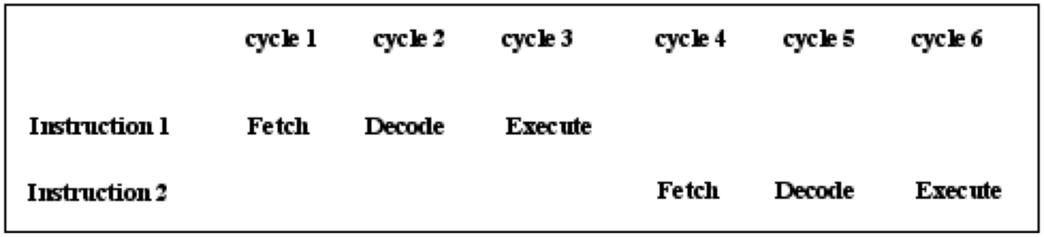
Bit (D-FF)

Fetch-Execution Cycle

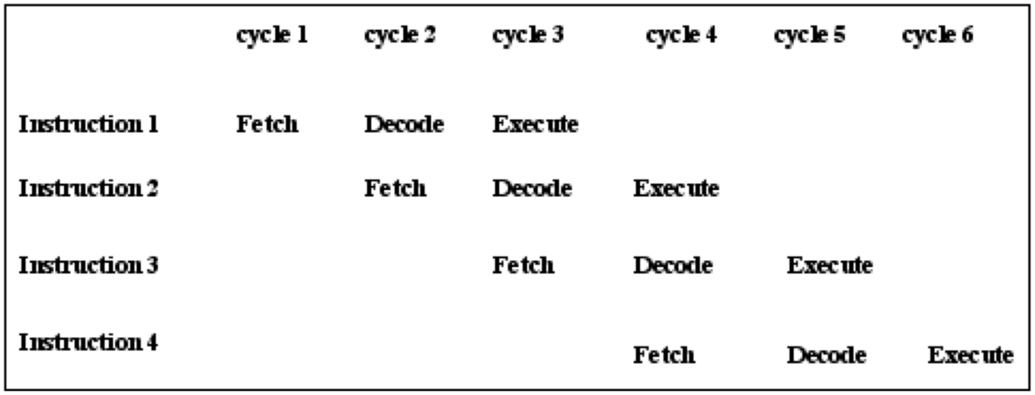
- Basic Operations of MCU
  - Fetch, Decode, Execute
- Two models
  - Sequential fetch-execute cycle
    - Complete the cycle before starting a new one
    - **Time to complete the task: T = t1+ t2 + t3 + ... + tn**
  - Pipelining
    - Break the fetch-execute cycle into a number of separate stages, so that when one stage is being carried out for a particular instruction, the CPU can carry out another stage for a second instruction, and so on.
    - Originated from the basic concept used in assembly lines
      - Each instruction still takes the same number of cycles to complete
      - The gain comes from the fact that the CPU can operate on instructions in the different stages in **parallel**.
      - The total time to complete the task is the same as above.
      - **Clock period for completing each task: Tp = max(t1, t2, t3, ..., tn)**

Pipelining

- Fetch-execute cycle using sequential vs. Fetch-execute cycle using pipelining
- **Throughput** of the operation is defined as 1/T (operations or instructions/second)
- Generally, the faster the clock the higher the throughput will be – however…. (next slide)



	cycle 1	cycle 2	cycle 3	cycle 4	cycle 5	cycle 6
Instruction 1	Fetch	Decode	Execute			
Instruction 2				Fetch	Decode	Execute



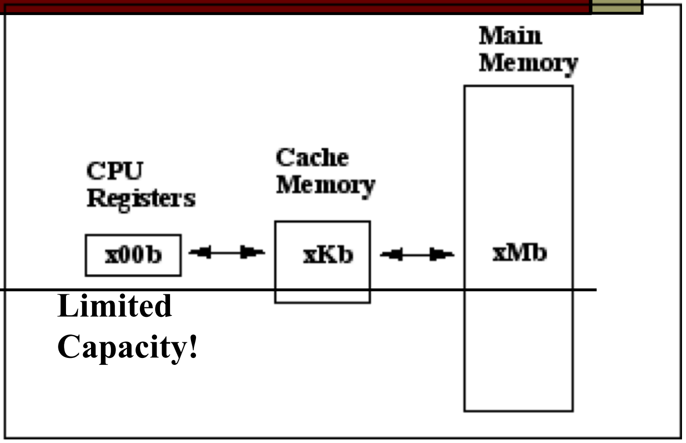
	cycle 1	cycle 2	cycle 3	cycle 4	cycle 5	cycle 6
Instruction 1	Fetch	Decode	Execute			
Instruction 2		Fetch	Decode	Execute		
Instruction 3			Fetch	Decode	Execute	
Instruction 4				Fetch	Decode	Execute

Assuming all stages are finished in a single (or n) clock cycle

Clock Rate Limitation in Pipelining

- Increasing the clock speed does **not guarantee** significant performance gains.
- This is because the speed of the processor is effectively determined by **the rate at which it can fetch instructions and data from memory.**
  - Example: if the processor spends 90% of its time waiting on memory, the performance gained by doubling the processor speed (without improving the memory access time) is only 5%. ← Make sure you get it!

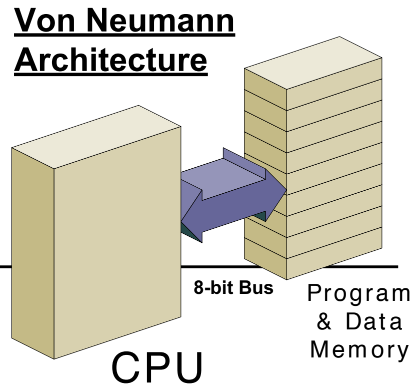
Cache



- One way of improving memory access time
  - use of a **cache memory system**
- The processor operates at its maximum speed if the data to be processed is in its registers.
  - Register storage capacity is very limited!
- One, very effective way of overcoming the slow access time of main memory, is to design a faster intermediate memory system, that lies between the CPU and main memory.
- Such memory is called **cache memory** (or simply **cache**)
  - Cache memory is high speed memory (e.g. SRAM) which can be accessed much more quickly than normal memory (usually dynamic RAM (DRAM)).
  - It has a **smaller capacity** than main memory and it holds recently accessed data from main memory
- Computers may use separate memories to store instructions and data
  - **Harvard Architecture**

http://www.csi.ucd.ie/staff/jcarthy/home/alp/alp7.html

Memory Model – Von Neumann Architecture

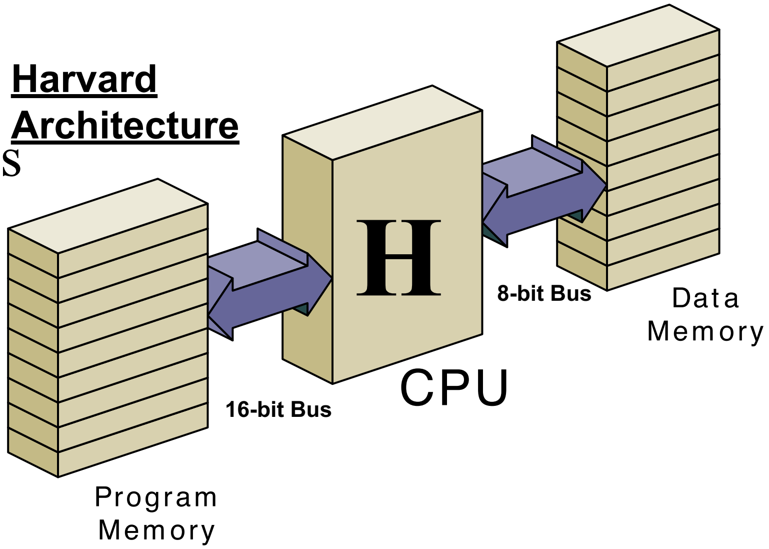


- Fetches instructions and data from a single memory space
  - Also known as Princeton architecture
- Limits operating bandwidth
- CISC designs are also more likely to feature this model
- Uses **unified cache** memory: instructions and data may be stored in the same cache memory
- Can be either reading an instruction or reading/writing data from/to the memory
  - Both cannot occur at the same time since the instructions and data use the same bus system.

http://en.wikipedia.org/wiki/Harvard_architecture

Memory Model –
(Pure or Strict) Harvard Architecture

- The original Harvard architecture computer, the Harvard Mark I, employed entirely separate memory systems to store instructions and data.
- Uses two separate memory spaces for program instructions and data - separate pathways with separate address spaces
  - Allows for different bus widths
  - Improved operating throughput



- RISC designs are also more likely to feature this model
- Note that having separate address spaces can create issues for high-level programming no supporting different address spaces (not good for CISC!)
- The CPU can both read an instruction and perform a data memory access at the same time, even without a cache
  - Faster (than Von Neumann) for a given circuit complexity because instruction fetches and data access do not contend for a single memory pathway.
- Example: PIC Microcontrollers (Separate code and data spaces)

Memory Model – (Modified or Non-Strict) Harvard architecture

- A Modified Harvard architecture machine is very much like a Harvard architecture machine
- Modification can be different
  1. The program and data memory occupy different address spaces, but there are operations to read and/or write program memory as data.
  2. It relaxes the strict separation between memories while still letting the CPU concurrently access two (or more) memory busses
     - It offers separate pathways with the unified address spaces of the memory
     - As far as the programmer is concerned the machine performs like a von Neumann machine
- Remember: many modern computers that are documented as Harvard Architecture are, in fact, Modified Harvard Architecture
- Applications
  - Atmel AVR 8-bit RISC microcontroller
  - PlayStation Portable's WLAN chip, and many more; anything with enhanced DSP application; x86 (Intel) processors, ARM cores (ARM9) embedded as applications processors in cell phones, and PowerPC.

Some Examples:

- Microcontrollers
  - LPC210x - ARM7 Microcontroller LPC210x – RISC-based microcontroller; Harvard
  - ATmega128 - AVR Microcontroller (developed by Atmel) , Harvard, RISC
  - PIC Microcontroller – Harvard, RISC
  - 68HC11/MC68HC24; descended from Motorola 68xxx microprocessor, which is a 8-bit CISC microcontroller - Von Neumann architecture
  - Z8 Microcontrollers – Harvard
  - Intel 8051 - 8-bit Harvard architecture, single chip microcontroller ; CISC instruction
- Microprocessors
  - Intel x86 – CICS; Von Neumann (Intel, AMD, etc.)
  - 68xxx Motorola - 16/32-bit CISC - competitor to Intelx86
  - ARM - 32 bit (used by Atmel) , RISC
  - SPARC V9 ISA – used in Sun UltraSparc – RISC processor; developed by Sun Microsystems

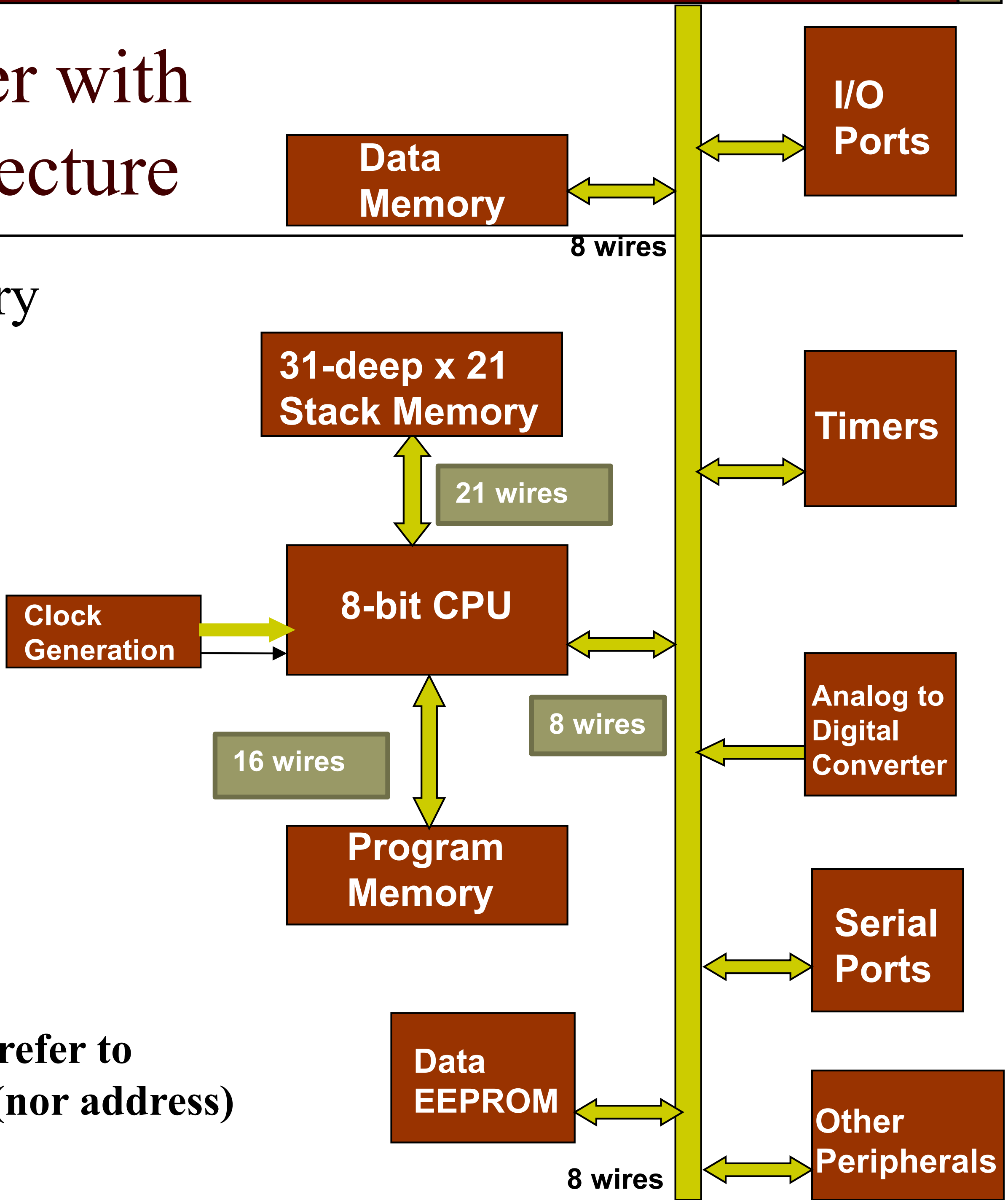
http://www.experiencefestival.com/microcontroller_-_intel

Main 8-bit Controllers

- Microchip
  - RISC architecture (reduced instruction set computer)
  - Has sold over 2 billion as of 2002
  - Cost effective and rich in peripherals
- Motorola
  - CISC architecture
  - Has hundreds of instructions
  - Examples: 68HC05, 68HC08, 68HC11
- Intel
  - CISC architecture
  - Has hundreds of instructions
  - Examples: 8051, 8052
  - Many difference manufacturers: Philips, Dallas/MAXIM Semiconductor, etc.
- Atmel
  - RISC architecture (reduced instruction set computer)
  - Cost effective and rich in peripherals
  - AVR

PIC Microcontroller with the Harvard Architecture

- Three types of memory
  - Data Memory
  - Program Memory
  - Stack Memory



Numbers refer to Data bus (nor address)

Program Memory

- Program memory is 16-bits wide accessed through a separate program data bus and address bus inside the PIC18.
- Program memory stores the program and also static data in the system.
  - On-chip
  - External
- On-chip program memory is either PROM or EEPROM.
  - The PROM version is called OTP (one-time programmable) (PIC18C)
  - The EEPROM version is called Flash memory (PIC18F).
- Maximum size for program memory is 2M
  - Program memory addresses are 21-bit address starting at location 0x000000

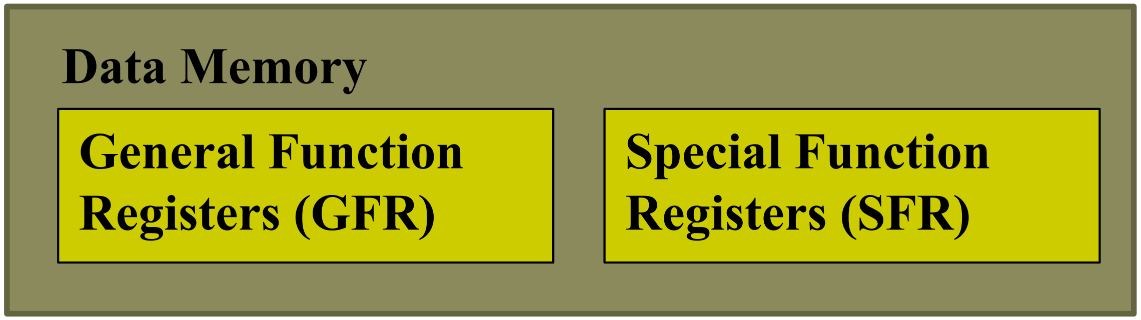
Example: PIC18F4520 has 32K program memory – draw it!

Data Memory (1)

- Used for **transitory** data when the program is being executed
  - Example: A=1, B=2, C=3 X=A+B+C→ A+B=W;W+C=W
- Data memory is either **SRAM or EEPROM**.
  - Some chips only have SRAM
  - Others may have SRAM and EEPROM
    - EEPROM stores permanently
- Various PIC18 versions contain between 256 and 3968 bytes of data memory
  - For example: SRAM data memory begins at 12-bit address 0x000 and ends at 12-bit address 0xFFF (4K)

Data Memory (2)

- Data memory is often divided into two sections
  - General Function Registers (GFR) or register file location
    - 000-0xF7F locations
  - Special Function Registers (SFR) – specific to PIC
    - 0xF80-0xFFF (upper 128 bytes)
- Depending on the PIC chip, the sizes for GFR and SFR are different



Data Memory

General Function Registers (GFR)

Special Function Registers (SFR)

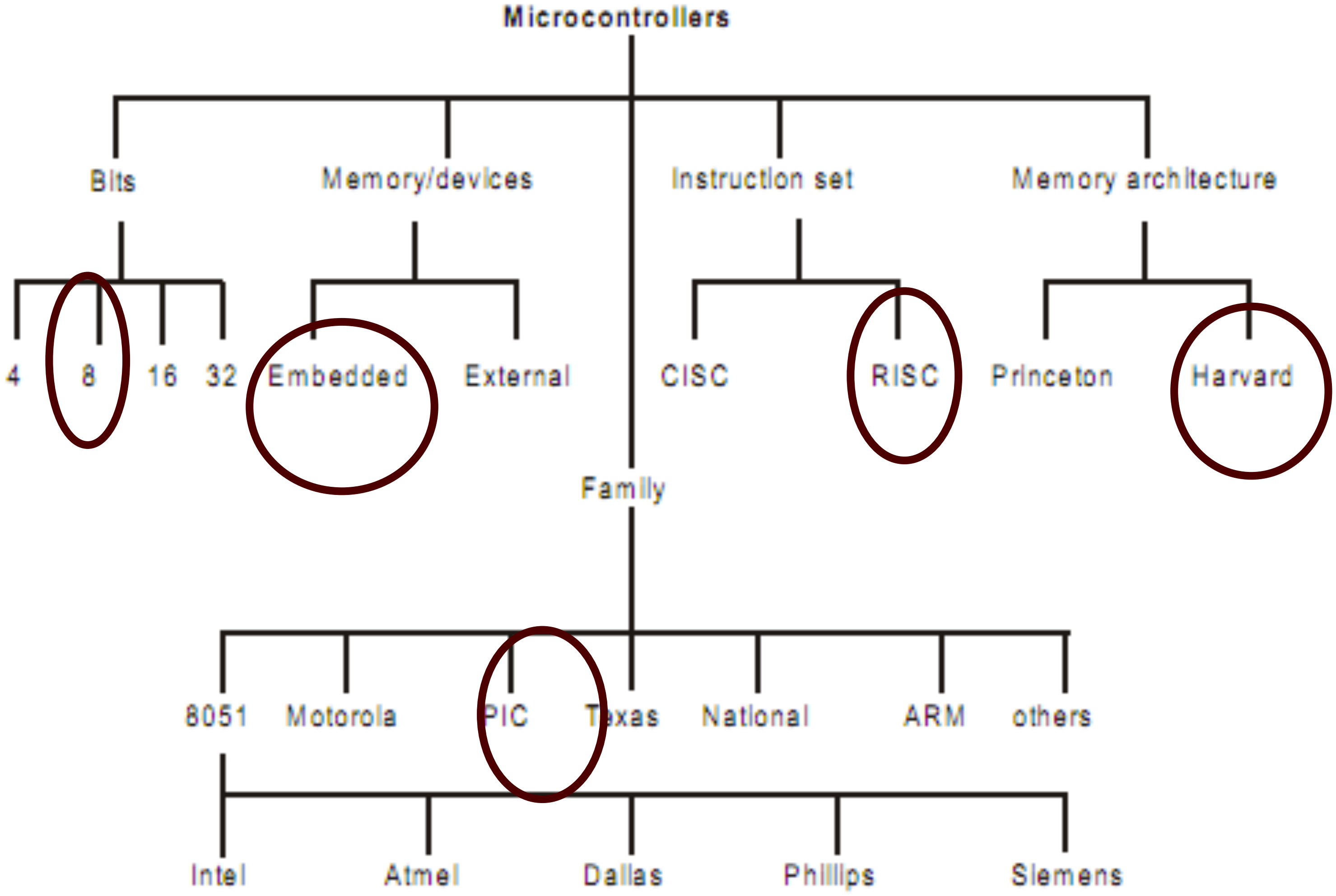
Program Stack Memory

Saving the return address

- The PIC18 contains a program stack that stores up to 31 return addresses from functions.
  - 31-deep
  - The program stack is 21 bits in width as is the program address (remember address memory is 2M)
- Stack memory uses SRAM
- Operation of a stack
  - When a function is called, the return address (location of the next step in a program) is pushed onto the stack.
    - For example: Stack number 1 will have value= 0x0x1F0000
  - When the return occurs within the function, the return address is retrieved from the stack and placed into the program counter.

Summary: Microcontroller Types

Our focus for the rest of the course:



http://www.newagepublishers.com/samplechapter/001599.pdf

Microchip Technology

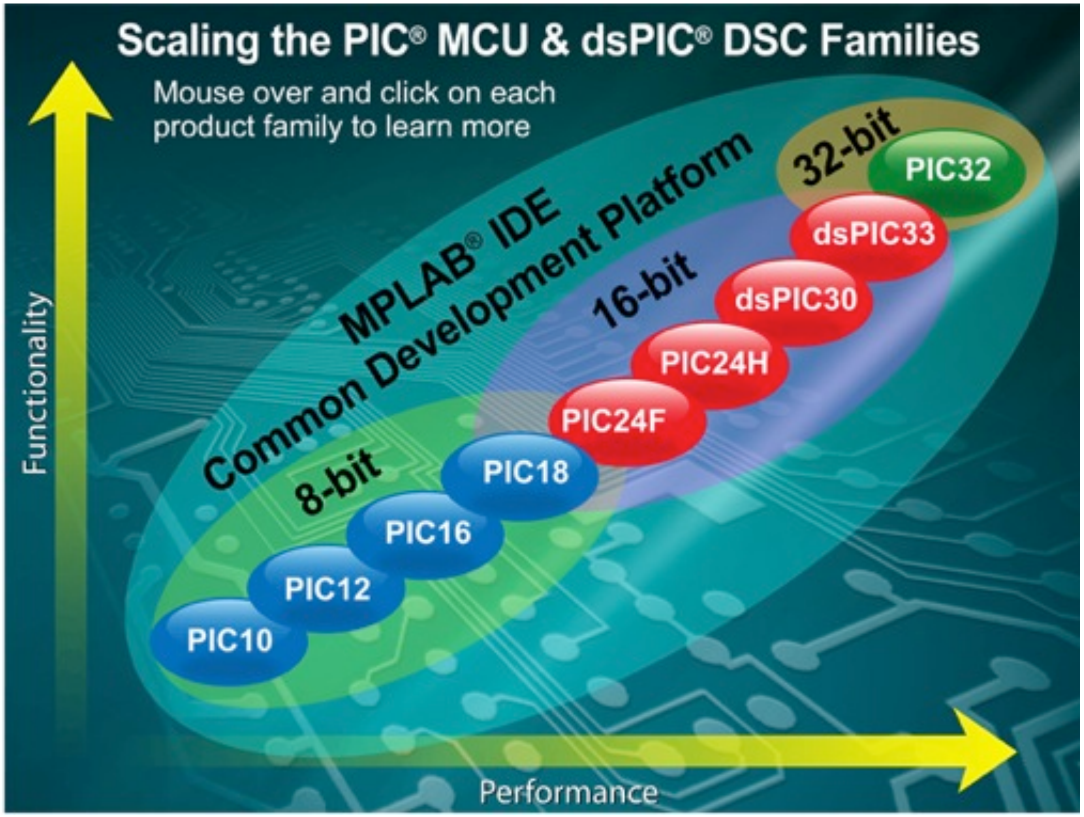
- PIC Microcontroller
  - **Programmable Interface Controller**
- 8-bit MCU (depending on RAM size, IO pins, stack size, enhanced architecture)
  - Base-line (including Dust, 6-pin, no interrupts)
  - Mid-range
  - High-end (PIC18F, uses C18 compiler)
- 16-bit MCU
  - Enhanced with dsp features (support for VoIP)
  - Smaller, faster, low-power; uses C30 Compiler
  - PIC24, dsp30/33
- 32-bit MCU
  - Instruction cache, low-power, faster RAM
  - C32 compiler

See this link:

http://www.microchip.com/pagehandler/en-us/family/8bit/#8bitVideoChannel

PIC18F452/4520/45K20 Memory - Example

- Program Memory: 32 K (2^{15})
  - Address range: 000000 to 007FFFH
  - 16-bit registers
- Data Memory: 4 K
  - Address range: 000 to FFFH
  - 8-bit registers
- Data EEPROM
  - Not part of the data memory space
  - Addressed through special function registers

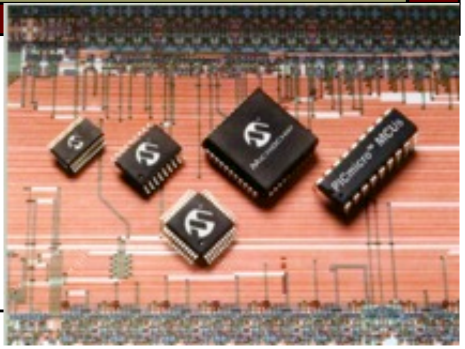


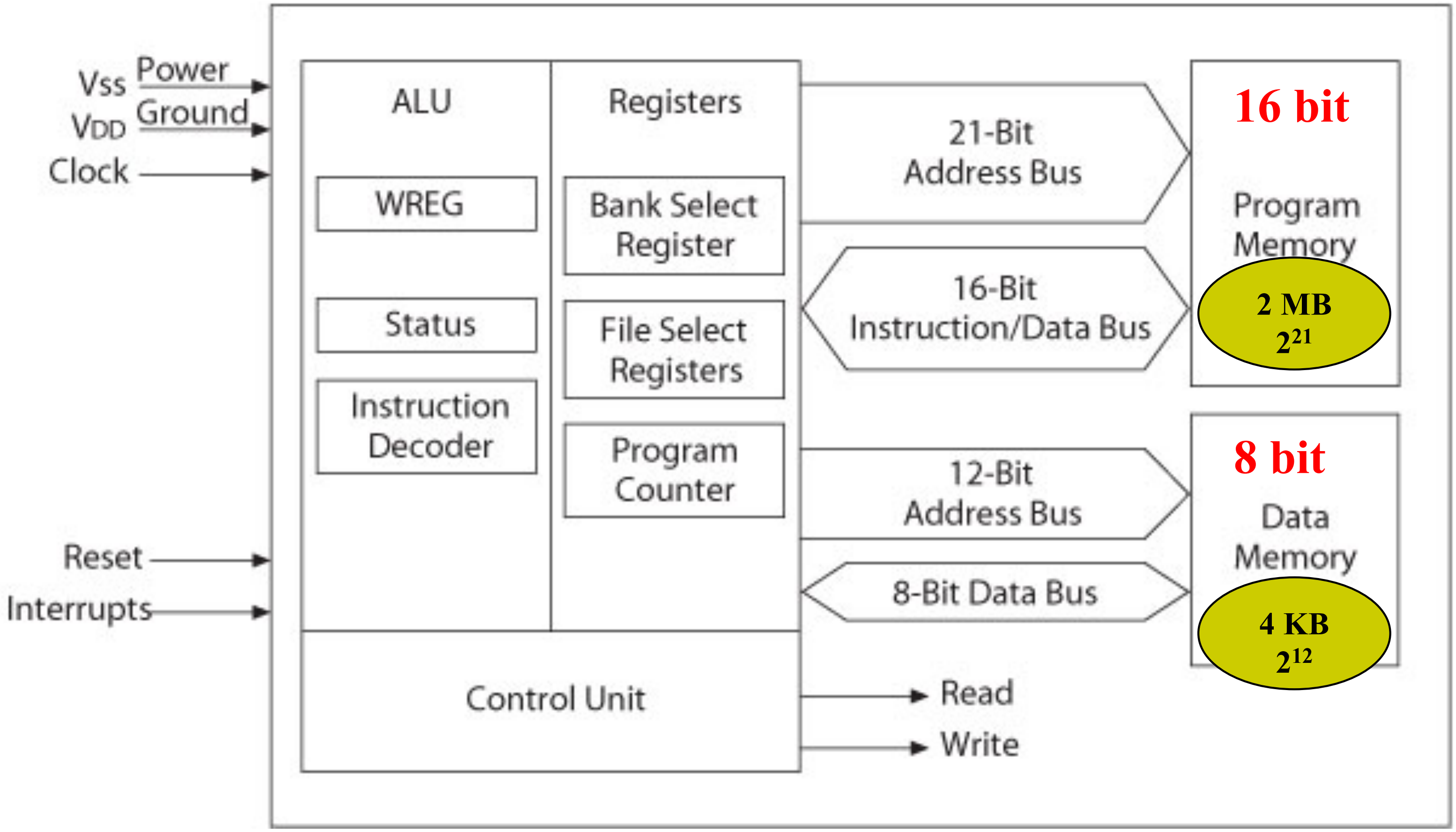
Scaling the PIC® MCU & dsPIC® DSC Families
Mouse over and click on each product family to learn more
Functionality
Performance
MPLAB® IDE
Common Development Platform
8-bit
16-bit
32-bit
PIC10
PIC12
PIC16
PIC18
PIC24F
PIC24H
dsPIC30
dsPIC33
PIC32

See this link:

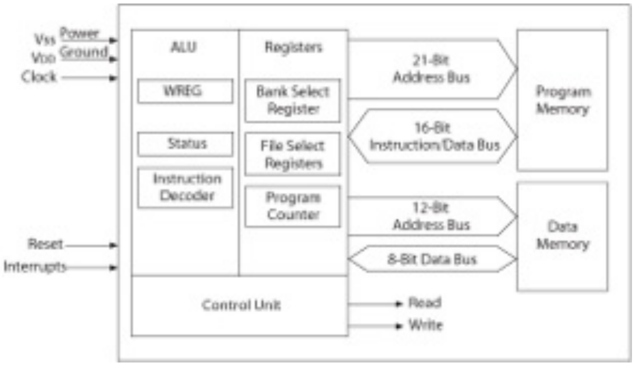
http://www.microchip.com/ParamChartSearch/chart.aspx?branchID=1004&mid=10&lang=en&pageId=74

PIC18F – MCU and Memory





PIC18F – MCU and Memory – Design Problem

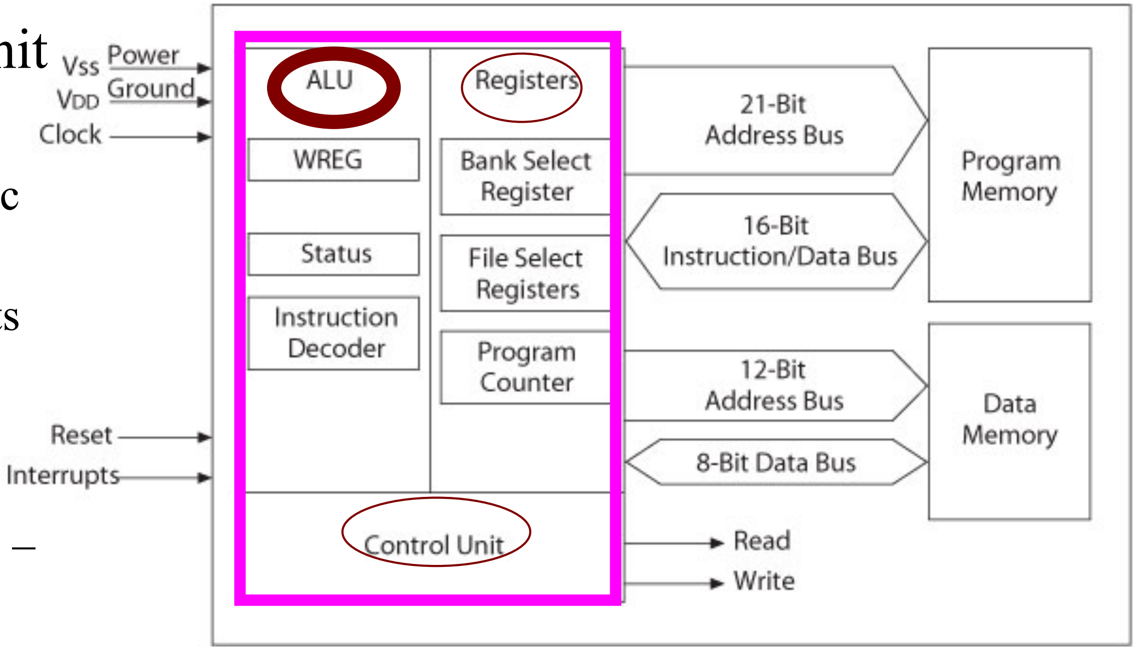


- Design a microcontroller with the following specifications Specify bus widths.
  - Program Memory: 32 K (15 bits)
  - Data Memory: 4 K (12 bits)
- In your design show where the counter registers are located
- In your design show where the working registers are located (which part of the microprocessor unit)
- Assuming each memory has a R/W and OE, show how they are connected to MPU
  - Show where the read/write lines are connected to – specify the direction of each.

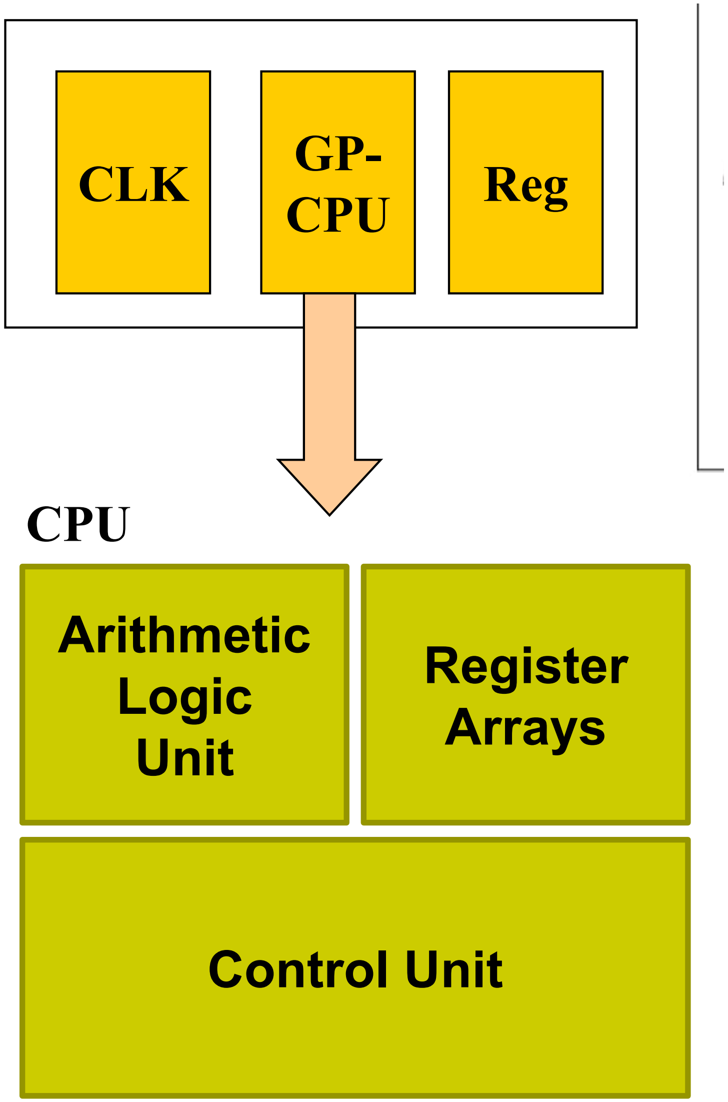
Do it on your own!

Microprocessor Unit (1 of 3)
Includes Arithmetic Logic Unit (ALU)

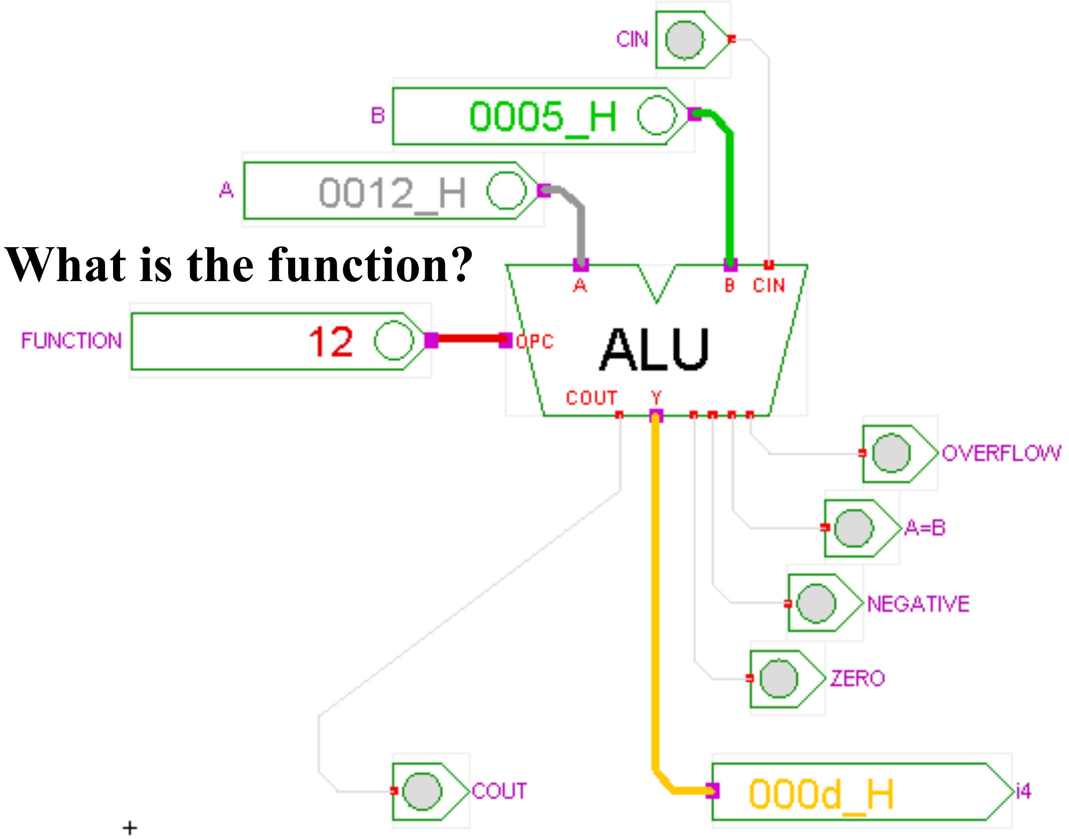
- Includes Arithmetic Logic Unit (ALU), Registers, and Control Unit
  - Arithmetic Logic Unit (ALU)
    - Performs logical and arithmetic functions
    - WREG – working register (acts as an accumulator) – used to perform arithmetic or logical functions
    - Status register that stores flags – indicates the status of the operation done by ALU
    - Instruction decoder (ID)– when the instruction is fetched it goes into the ID to be interpreted – tell the processor what to do



Microprocessor Unit (1 of 3)
Includes Arithmetic Logic Unit (ALU)
General ALU Architecture



All arithmetic and logical instructions are carried out by the ALU.



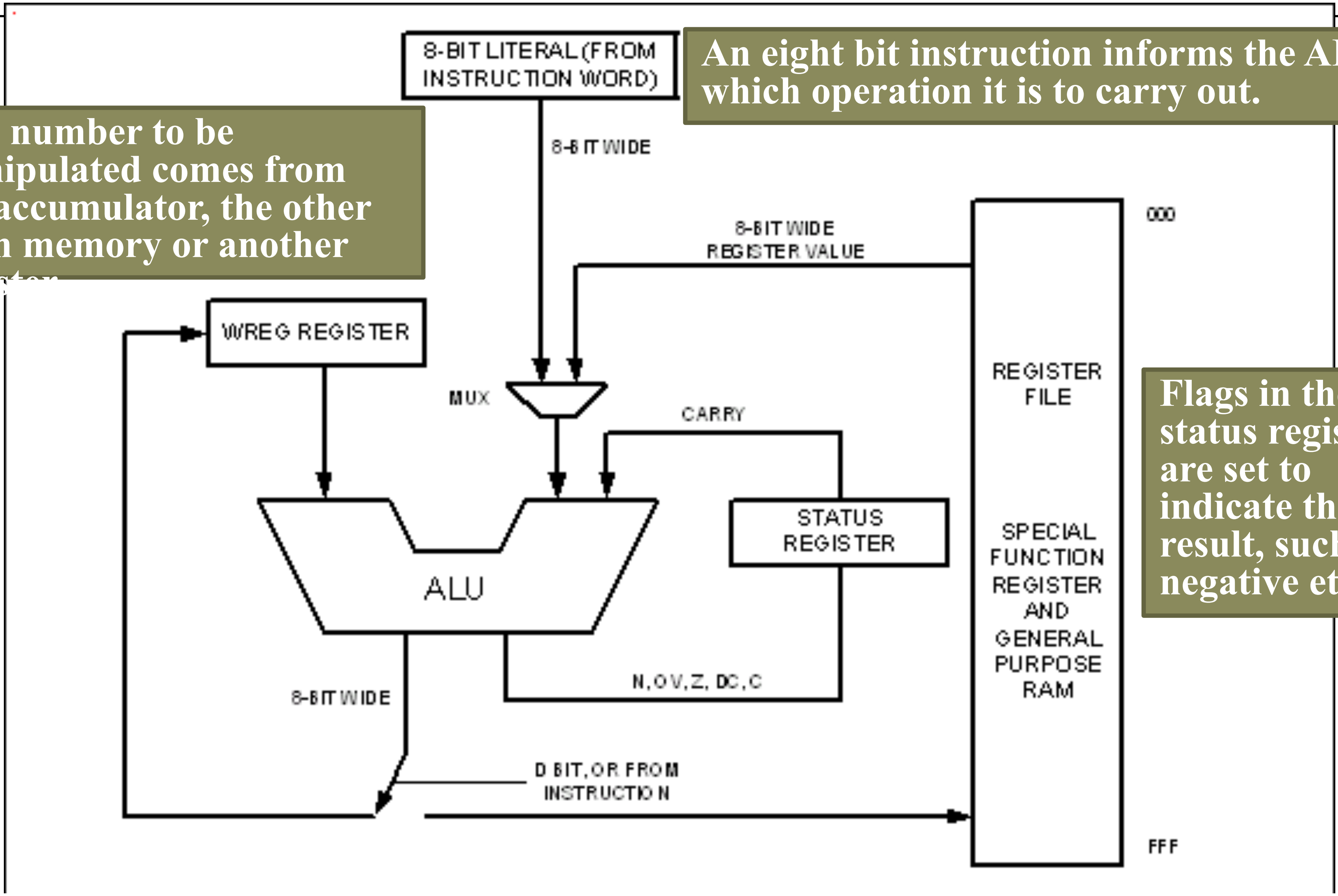
What is the function?

http://tams-www.informatik.uni-hamburg.de/applets/hades/webdemos/50-rtlib/50-alu/alu.html

Microprocessor Unit (1 of 3)

Includes Arithmetic Logic Unit (ALU)

General ALU Architecture


8-BIT LITERAL (FROM INSTRUCTION WORD)
8-BIT WIDE
8-BIT WIDE REGISTER VALUE
000
WREG REGISTER
MUX
CARRY
REGISTER FILE
STATUS REGISTER
ALU
SPECIAL FUNCTION REGISTER AND GENERAL PURPOSE RAM
N, OV, Z, DC, C
8-BIT WIDE
D BIT, OR FROM INSTRUCTION
FFF

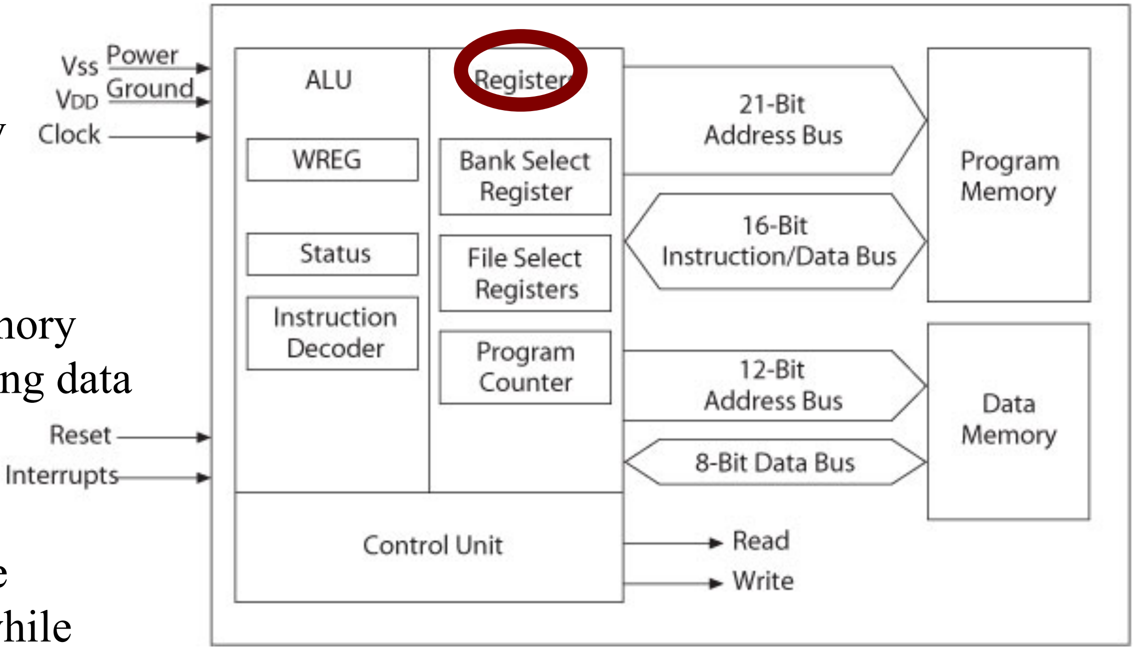
An eight bit instruction informs the ALU which operation it is to carry out.

One number to be manipulated comes from the accumulator, the other from memory or another register

Flags in the status register are set to indicate the result, such as negative etc

Microprocessor Unit (2 of 3)

- Registers – hold memory address
  - Bank Select Register (BSR)
    - 4-bit register used in direct addressing the data memory
  - File Select Registers (FSRs)
    - 16-bit registers used as memory pointers in indirect addressing data memory
  - Program Counter (PC)
    - 21-bit register that holds the program memory address while executing programs



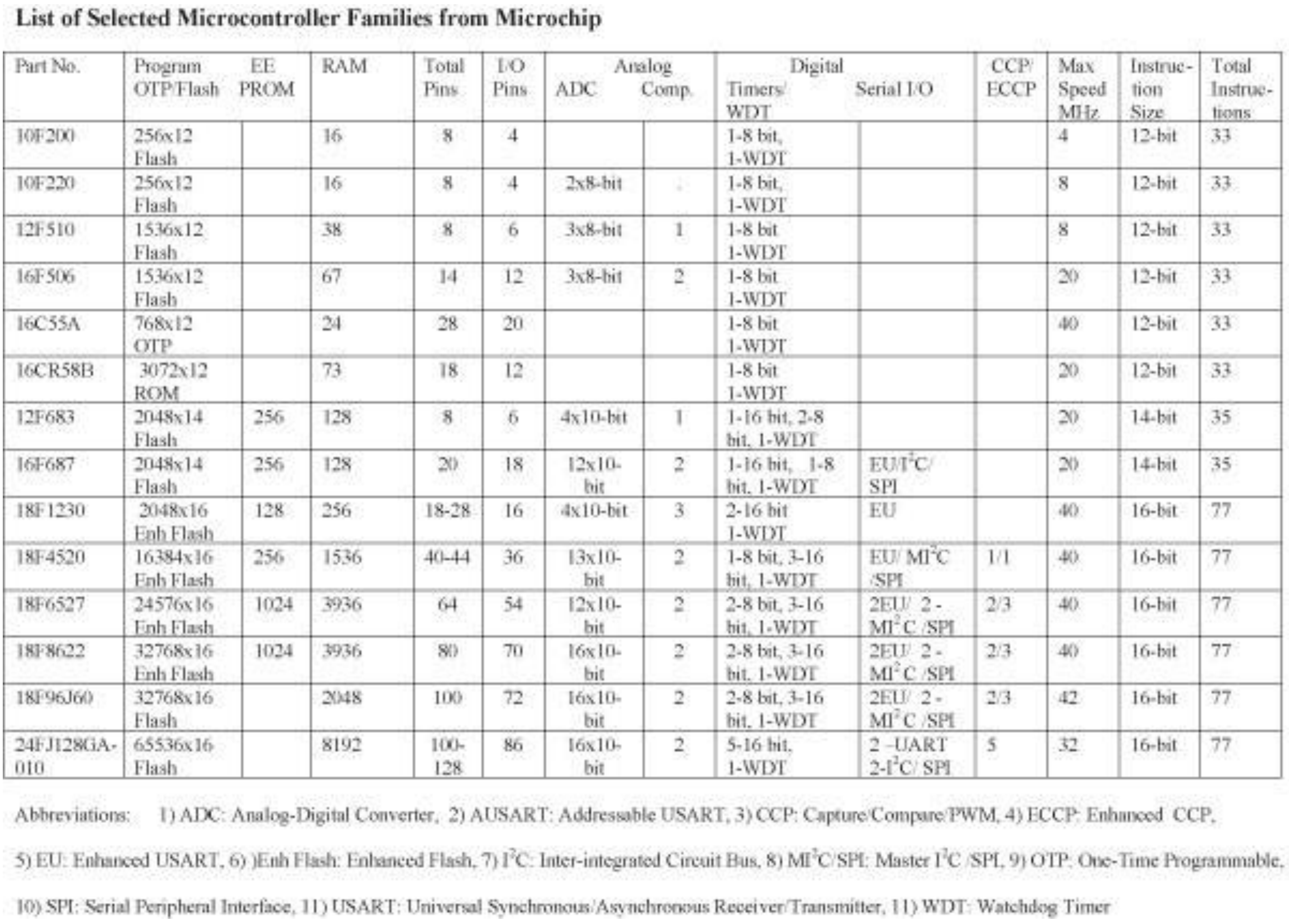
Microprocessor Unit (3 of 3)

- Control unit
  - Provides timing and control signals to various Read and Write operations

Examples

- Show how four 8-bit RAM blocks each having 1KB capacity can be connected to the CPU. Assume the address bus is limited to 10 bits and each RAM chip has the following pins: OE and R/W

Refer to your notes: Using DEMUX

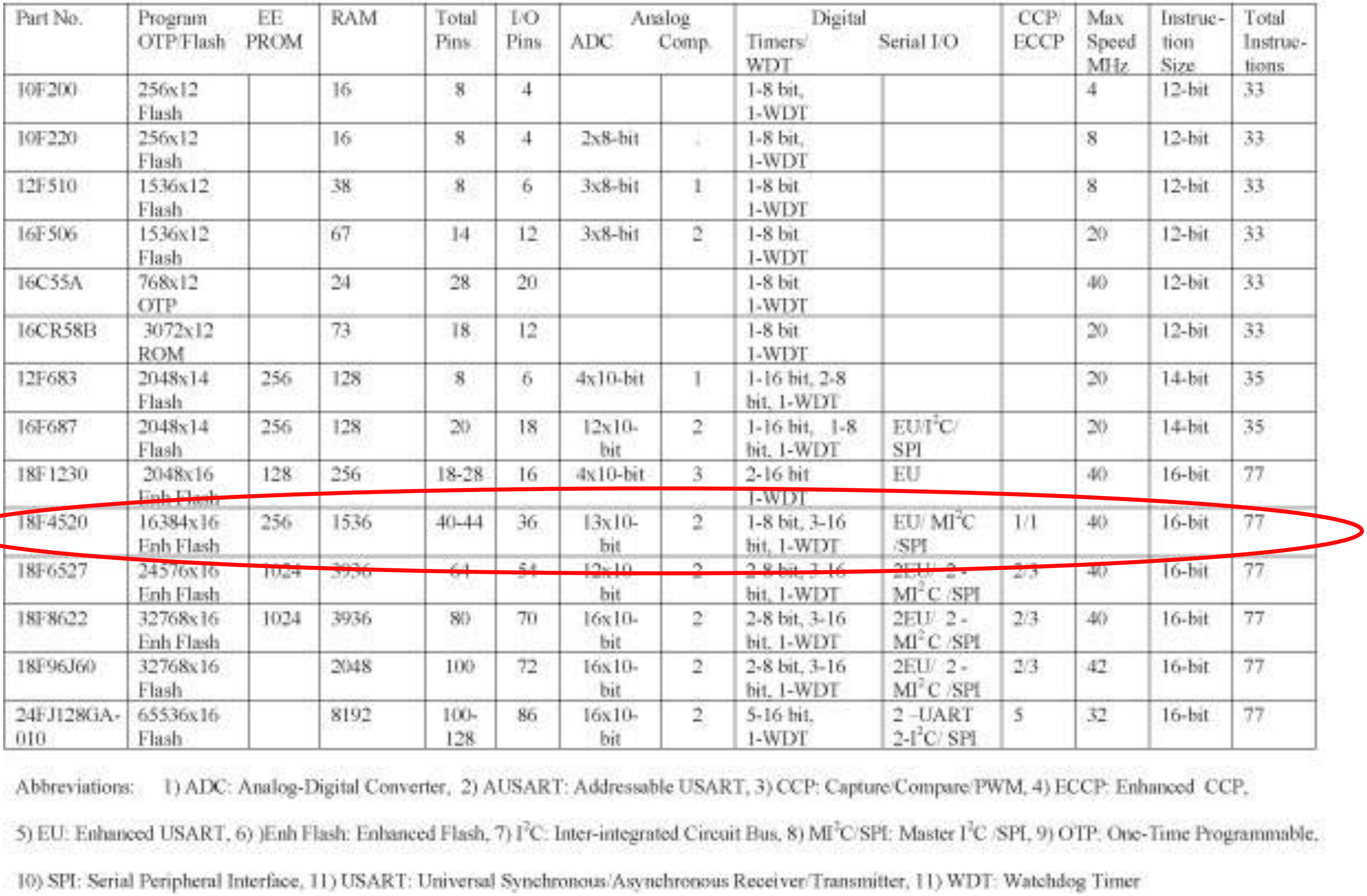


List of Selected Microcontroller Families from Microchip

Part No.	Program OTP/Flash	EE PROM	RAM	Total Pins	I/O Pins	Analog		Digital		CCP/ ECCP	Max Speed MHz	Instruction Size	Total Instructions
						ADC	Comp.	Timers/ WDT	Serial I/O				
10F200	256x12 Flash		16	8	4			1-8 bit, 1-WDT			4	12-bit	33
10F220	256x12 Flash		16	8	4	2x8-bit	-	1-8 bit, 1-WDT			8	12-bit	33
12F510	1536x12 Flash		38	8	6	3x8-bit	1	1-8 bit 1-WDT			8	12-bit	33
16F506	1536x12 Flash		67	14	12	3x8-bit	2	1-8 bit 1-WDT			20	12-bit	33
16C55A	768x12 OTP		24	28	20			1-8 bit 1-WDT			40	12-bit	33
16CR58B	3072x12 ROM		73	18	12			1-8 bit 1-WDT			20	12-bit	33
12F683	2048x14 Flash	256	128	8	6	4x10-bit	1	1-16 bit, 2-8 bit, 1-WDT			20	14-bit	35
16F687	2048x14 Flash	256	128	20	18	12x10-bit	2	1-16 bit, 1-8 bit, 1-WDT	EU/I^{2}C/ SPI		20	14-bit	35
18F1230	2048x16 Enh Flash	128	256	18-28	16	4x10-bit	3	2-16 bit 1-WDT	EU		40	16-bit	77
18F4520	16384x16 Enh Flash	256	1536	40-44	36	13x10-bit	2	1-8 bit, 3-16 bit, 1-WDT	EU/ MI^{2}C /SPI	1/1	40	16-bit	77
18F6527	24576x16 Enh Flash	1024	3936	64	54	12x10-bit	2	2-8 bit, 3-16 bit, 1-WDT	2EU/ 2 - MI^{2}C /SPI	2/3	40	16-bit	77
18F8622	32768x16 Enh Flash	1024	3936	80	70	16x10-bit	2	2-8 bit, 3-16 bit, 1-WDT	2EU/ 2 - MI^{2}C /SPI	2/3	40	16-bit	77
18F96J60	32768x16 Flash		2048	100	72	16x10-bit	2	2-8 bit, 3-16 bit, 1-WDT	2EU/ 2 - MI^{2}C /SPI	2/3	42	16-bit	77
24FJ128GA-010	65536x16 Flash		8192	100-128	86	16x10-bit	2	5-16 bit, 1-WDT	2 –UART 2-I^{2}C/ SPI	5	32	16-bit	77

Abbreviations: 1) ADC: Analog-Digital Converter, 2) AUSART: Addressable USART, 3) CCP: Capture/Compare/PWM, 4) ECCP: Enhanced CCP, 5) EU: Enhanced USART, 6))Enh Flash: Enhanced Flash, 7) I^{2}C: Inter-integrated Circuit Bus, 8) MI^{2}C/SPI: Master I^{2}C /SPI, 9) OTP: One-Time Programmable, 10) SPI: Serial Peripheral Interface, 11) USART: Universal Synchronous/Asynchronous Receiver/Transmitter, 11) WDT: Watchdog Timer

List of Selected Microcontroller Families from Microchip



Part No.	Program OTP/Flash	EE PROM	RAM	Total Pins	I/O Pins	Analog		Digital		CCP/ ECCP	Max Speed MHz	Instruc-tion Size	Total Instruc-tions
						ADC	Comp.	Timers/ WDT	Serial I/O				
10F200	256x12 Flash		16	8	4			1-8 bit, 1-WDT			4	12-bit	33
10F220	256x12 Flash		16	8	4	2x8-bit		1-8 bit, 1-WDT			8	12-bit	33
12F510	1536x12 Flash		38	8	6	3x8-bit	1	1-8 bit 1-WDT			8	12-bit	33
16F506	1536x12 Flash		67	14	12	3x8-bit	2	1-8 bit 1-WDT			20	12-bit	33
16C55A	768x12 OTP		24	28	20			1-8 bit 1-WDT			40	12-bit	33
16CR58B	3072x12 ROM		73	18	12			1-8 bit 1-WDT			20	12-bit	33
12F683	2048x14 Flash	256	128	8	6	4x10-bit	1	1-16 bit, 2-8 bit, 1-WDT			20	14-bit	35
16F687	2048x14 Flash	256	128	20	18	12x10-bit	2	1-16 bit, 1-8 bit, 1-WDT	EU/I^{2}C/ SPI		20	14-bit	35
18F1230	2048x16 Enh Flash	128	256	18-28	16	4x10-bit	3	2-16 bit 1-WDT	EU		40	16-bit	77
18F4520	16384x16 Enh Flash	256	1536	40-44	36	13x10-bit	2	1-8 bit, 3-16 bit, 1-WDT	EU/ MI^{2}C /SPI	1/1	40	16-bit	77
18F6527	24576x16 Enh Flash	1024	3936	64	54	12x10-bit	2	2-8 bit, 3-16 bit, 1-WDT	2EU/ 2 - MI^{2}C /SPI	2/3	40	16-bit	77
18F8622	32768x16 Enh Flash	1024	3936	80	70	16x10-bit	2	2-8 bit, 3-16 bit, 1-WDT	2EU/ 2 - MI^{2}C /SPI	2/3	40	16-bit	77
18F96J60	32768x16 Flash		2048	100	72	16x10-bit	2	2-8 bit, 3-16 bit, 1-WDT	2EU/ 2 - MI^{2}C /SPI	2/3	42	16-bit	77
24FJ128GA-010	65536x16 Flash		8192	100-128	86	16x10-bit	2	5-16 bit, 1-WDT	2 –UART 2-I^{2}C/ SPI	5	32	16-bit	77

Abbreviations: 1) ADC: Analog-Digital Converter, 2) AUSART: Addressable USART, 3) CCP: Capture/Compare/PWM, 4) ECCP: Enhanced CCP, 5) EU: Enhanced USART, 6))Enh Flash: Enhanced Flash, 7) I^{2}C: Inter-integrated Circuit Bus, 8) MI^{2}C/SPI: Master I^{2}C /SPI, 9) OTP: One-Time Programmable, 10) SPI: Serial Peripheral Interface, 11) USART: Universal Synchronous/Asynchronous Receiver/Transmitter, 11) WDT: Watchdog Timer

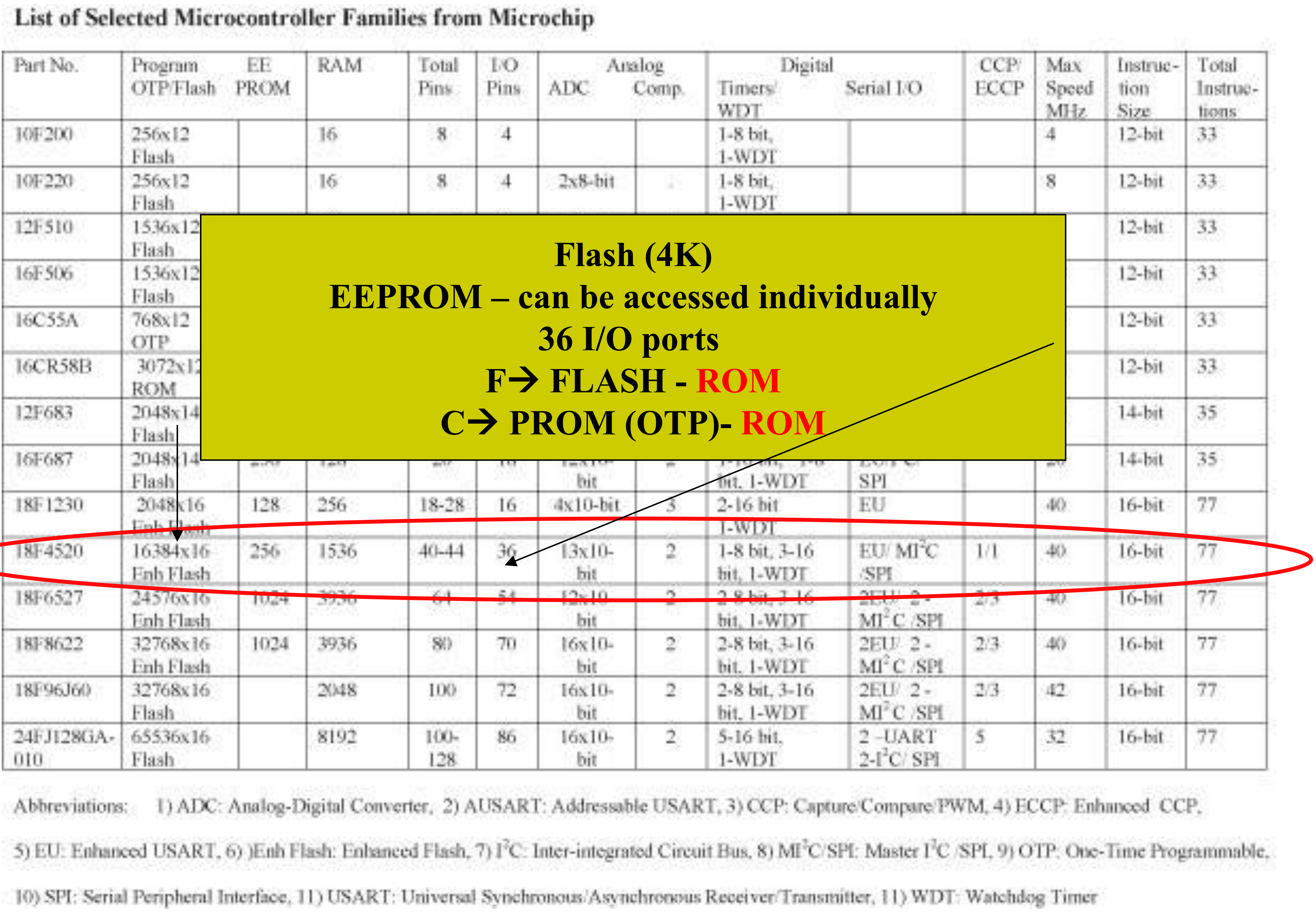
Flash (4K)

EEPROM – can be accessed individually

36 I/O ports

F→ FLASH - ROM

C→ PROM (OTP)- ROM



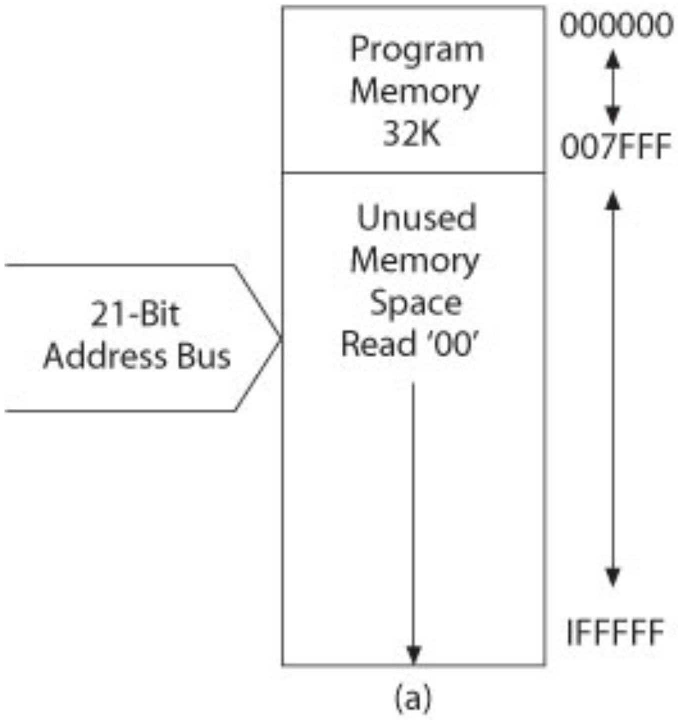
List of Selected Microcontroller Families from Microchip

Part No.	Program OTP/Flash	EE PROM	RAM	Total Pins	I/O Pins	Analog		Digital		CCP/ ECCP	Max Speed MHz	Instruction Size	Total Instructions
						ADC	Comp.	Timers/ WDT	Serial I/O				
10F200	256x12 Flash		16	8	4			1-8 bit, 1-WDT			4	12-bit	33
10F220	256x12 Flash		16	8	4	2x8-bit	-	1-8 bit, 1-WDT			8	12-bit	33
12F510	1536x12 Flash	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	12-bit	33
16F506	1536x12 Flash	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	12-bit	33
16C55A	768x12 OTP	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	12-bit	33
16CR58B	3072x1 ROM	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	12-bit	33
12F683	2048x14 Flash	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	14-bit	35
16F687	2048x14 Flash	[illegible]	[illegible]	[illegible]	[illegible]	[illegible] bit	[illegible]	[illegible] bit, 1-WDT	[illegible] SPI		[illegible]	14-bit	35
18F1230	2048x16 Enh Flash	128	256	18-28	16	4x10-bit	3	2-16 bit 1-WDT	EU		40	16-bit	77
18F4520	16384x16 Enh Flash	256	1536	40-44	36	13x10-bit	2	1-8 bit, 3-16 bit, 1-WDT	EU/ MI^{2}C /SPI	1/1	40	16-bit	77
18F6527	24576x16 Enh Flash	1024	3936	64	54	12x10-bit	2	2-8 bit, 3-16 bit, 1-WDT	2EU/ 2 - MI^{2}C /SPI	2/3	40	16-bit	77
18F8622	32768x16 Enh Flash	1024	3936	80	70	16x10-bit	2	2-8 bit, 3-16 bit, 1-WDT	2EU/ 2 - MI^{2}C /SPI	2/3	40	16-bit	77
18F96J60	32768x16 Flash		2048	100	72	16x10-bit	2	2-8 bit, 3-16 bit, 1-WDT	2EU/ 2 - MI^{2}C /SPI	2/3	42	16-bit	77
24FJ128GA-010	65536x16 Flash		8192	100-128	86	16x10-bit	2	5-16 bit, 1-WDT	2 –UART 2-I^{2}C/ SPI	5	32	16-bit	77

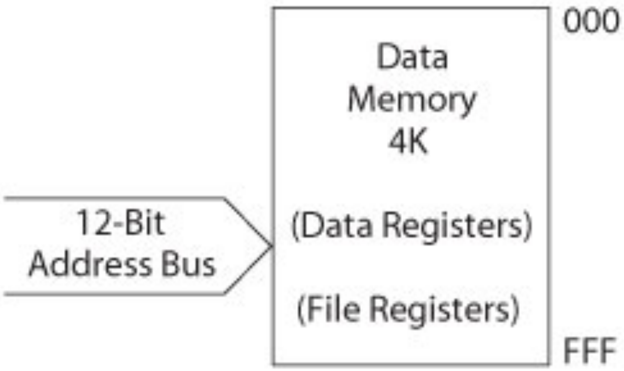
Abbreviations: 1) ADC: Analog-Digital Converter, 2) AUSART: Addressable USART, 3) CCP: Capture/Compare/PWM, 4) ECCP: Enhanced CCP, 5) EU: Enhanced USART, 6))Enh Flash: Enhanced Flash, 7) I^{2}C: Inter-integrated Circuit Bus, 8) MI^{2}C/SPI: Master I^{2}C /SPI, 9) OTP: One-Time Programmable, 10) SPI: Serial Peripheral Interface, 11) USART: Universal Synchronous/Asynchronous Receiver/Transmitter, 11) WDT: Watchdog Timer

PIC18F452/4520 Memory

- Program memory with addresses (Flash)-



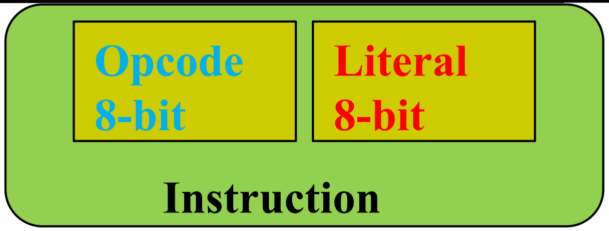
- Data memory with addresses
- Also called Data Register or File Register



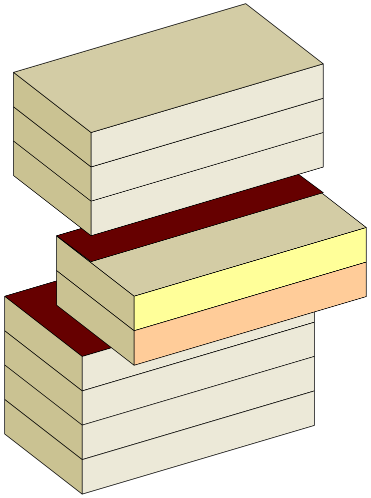
FFF=2^{12}=16x256=4096=4K

Remember: all instructions in PIC18 family are one word in length – read by the processor in one cycle;

Instructions



8-bit Program Memory



8-bit Instruction on typical 8-bit MCU

Example: Freescale ‘Load Accumulator A’:

- **2 Program Memory Locations**
- **2 Instruction Cycles to Execute**

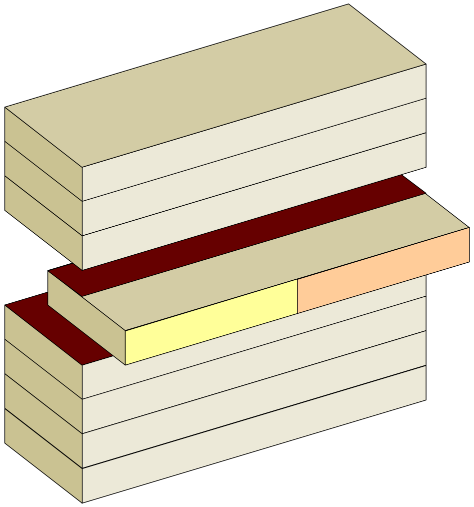
`inst k`

1	0	0	0	0	1	1	0
k	k	k	k	k	k	k	k

□ Limits Bandwidth

□ Increases Memory Size Requirements

16-bit Program Memory



16-bit Instruction on PIC18 8-bit MCU

Example: ‘Move Literal to Working Register’

- **1 Program Memory Location**
- **1 Instruction Cycle to Execute**

`movlw k`

0	0	0	0	1	1	1	0	k	k	k	k	k	k	k	k

● **Separate busses allow different widths**

● **2k x 16 is roughly equivalent to 4k x 8**

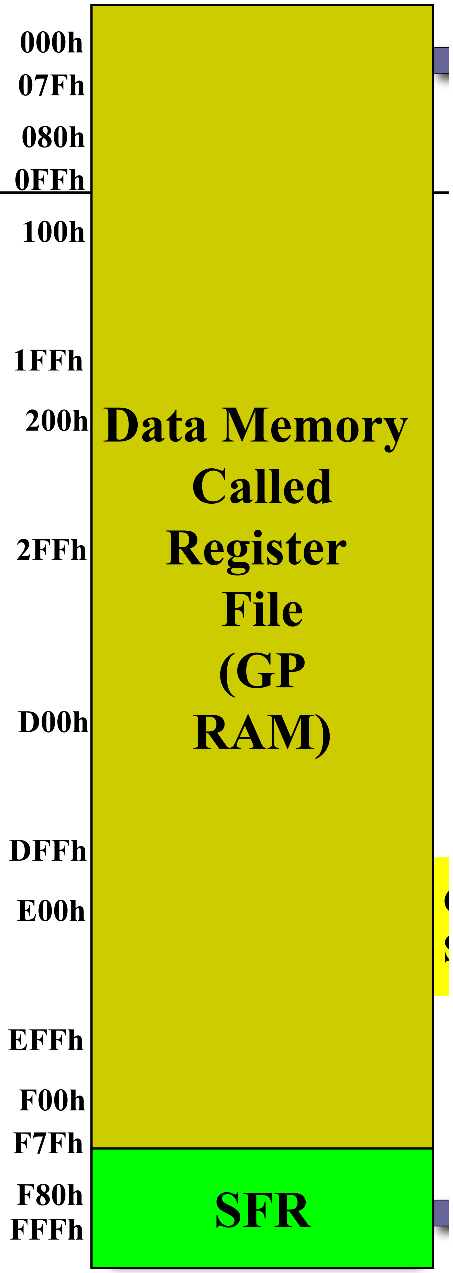
Direct and Indirect Addressing

- Direct addressing
  - MOVWF REG10 ; Directly writing W→ REG10
- Indirect addressing
  - We don't directly access the register by its address
  - We use pointers to access registers
  - For example, FSR0 contains the pointer value
    - We move W→FSR0 (special register)
    - Then the value stored in W will go into the register identified by FSR0

Data Memory Organization

FFF=2^{12}=16x256=4096=4KB SRAM

- Data Memory up to 4k bytes
  - **Data register map - with 12-bit address bus 000-FFF**

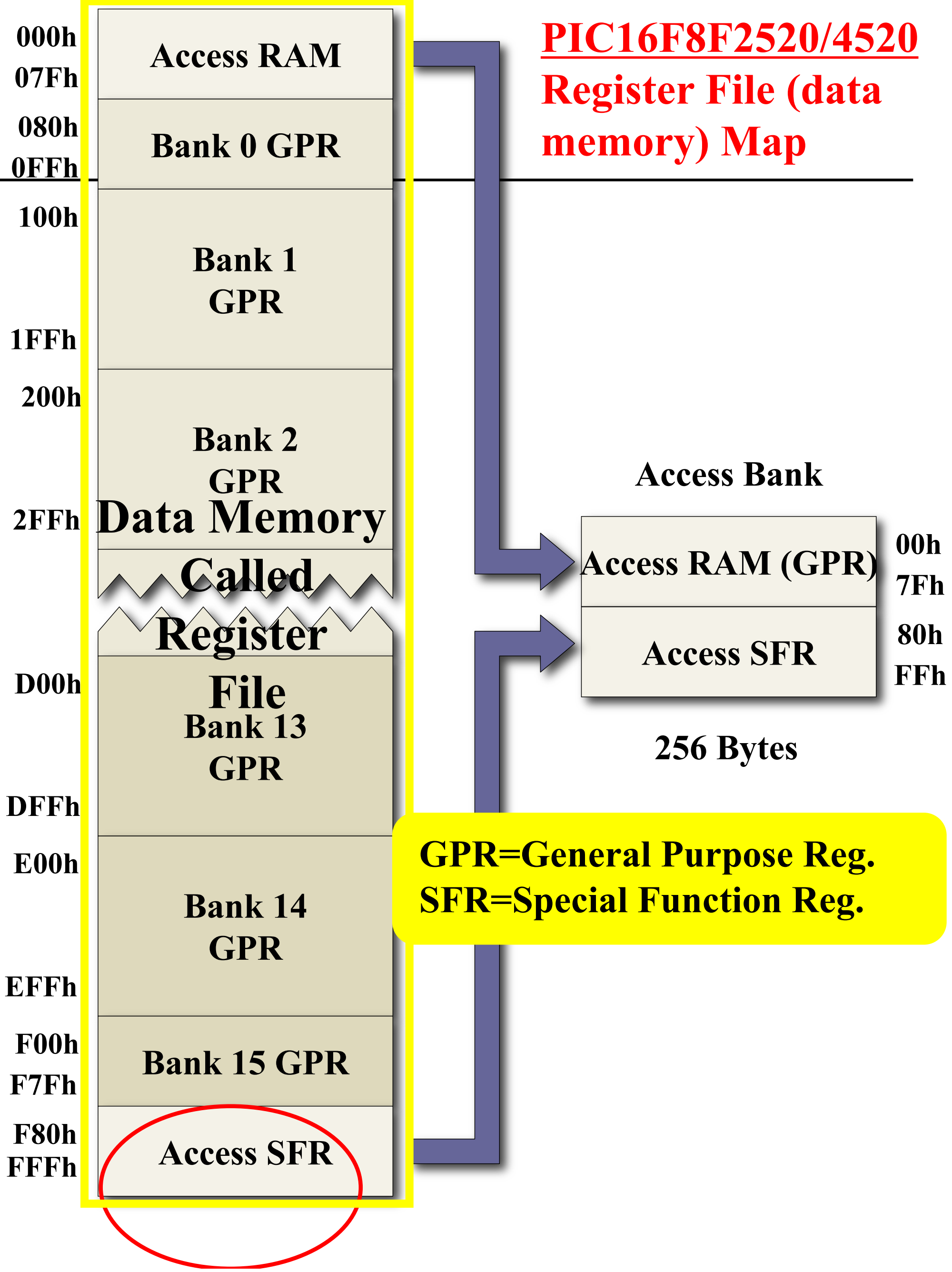


Data Memory Organization

FFF=2^{12}=16x256=4096=4KB SRAM

- Data Memory up to 4k bytes
  - **Data register map - with 12-bit address bus 000-FFF**
- Divided into 256-byte banks
- There are total of F banks
- Half of bank 0 and half of bank 15 form a virtual (or access) bank that is accessible no matter which bank is selected – this selection is done via 8-bits
  - Access Bank
    - SFR: Special Function Register (e.g., accessing the IO ports)
    - GPR: Used as a general purpose register

PIC16F8F2520/4520 Register File (data memory) Map

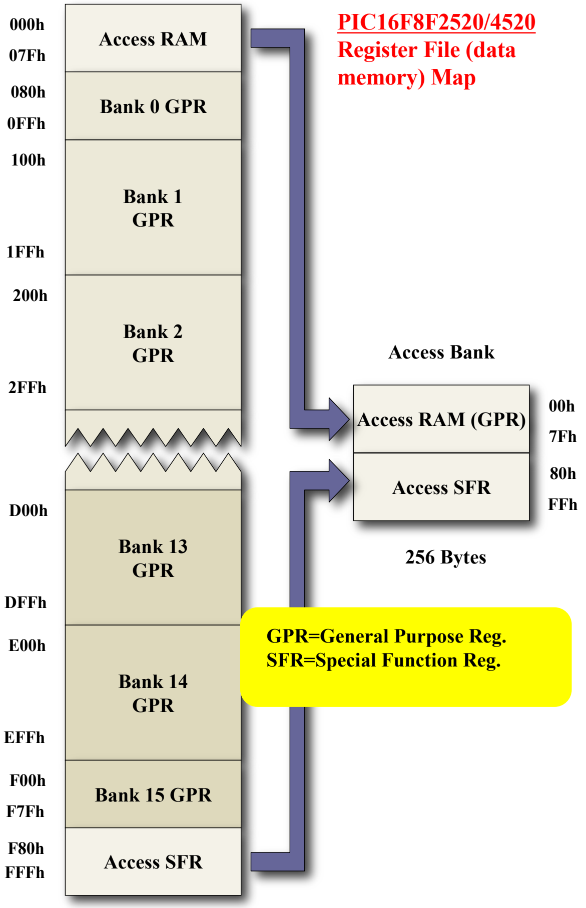


GPR=General Purpose Reg.
SFR=Special Function Reg.

PIC18F452/4520 – Data Memory with Access Banks

- Three ways to access data registers from the MPU:
  - Direct using Bank Select Registers (BSR)
    - Bank address (4-bit) + Instruction (8-bit)
  - Indirect using File Select Registers (FSR)
    - FSR contains the address of the data register
    - MPU uses FSR to access data registers
  - Access Bank
    - Directly accessible via 8-bits of register

Don' t confuse FSR and SFR!

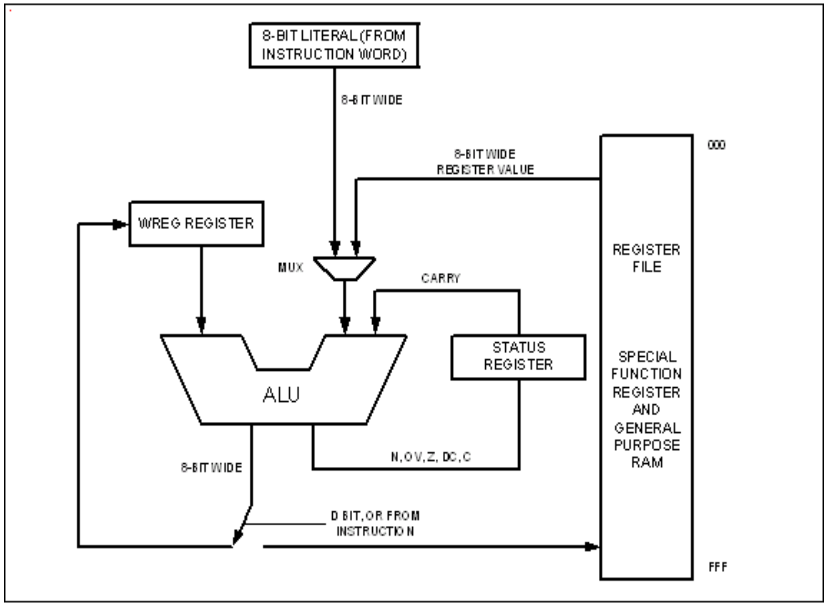


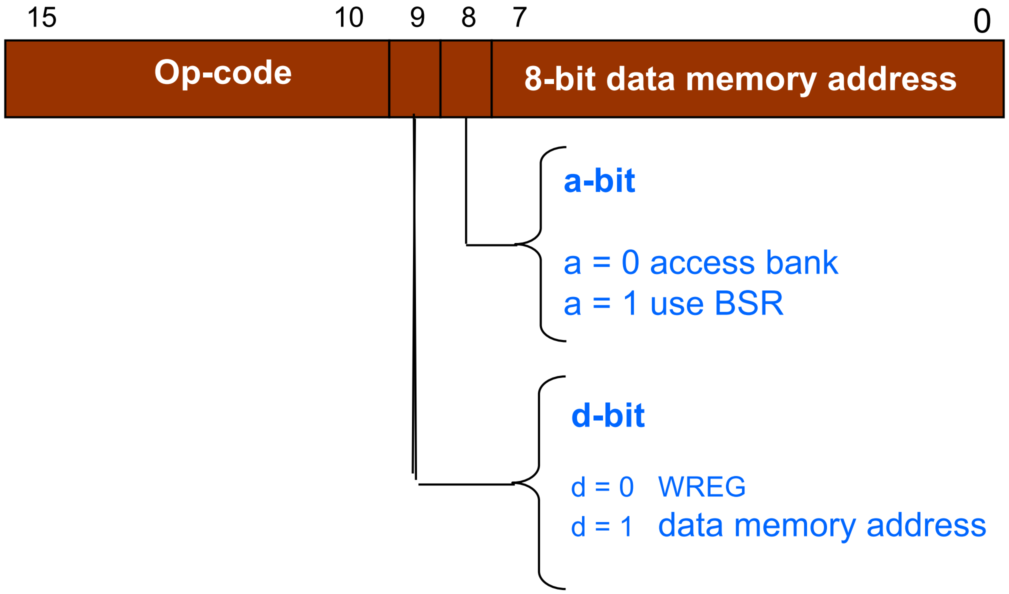
PIC16F8F2520/4520 Register File (data memory) Map

GPR=General Purpose Reg.
SFR=Special Function Reg.

So how do we know, say, address 0xF4 is referring to a SFR or GPR in BANK 0?

Basic Programming Model


8-BIT LITERAL (FROM INSTRUCTION WORD)
8-BIT WIDE
8-BIT WIDE REGISTER VALUE
000
WREG REGISTER
MUX
CARRY
REGISTER FILE
STATUS REGISTER
ALU
SPECIAL FUNCTION REGISTER AND GENERAL PURPOSE RAM
N,OV,Z, DC,C
8-BIT WIDE
D BIT, OR FROM INSTRUCTION
FFF



15 10 9 8 7 0

Op-code | | | 8-bit data memory address

a-bit

a = 0 access bank
a = 1 use BSR

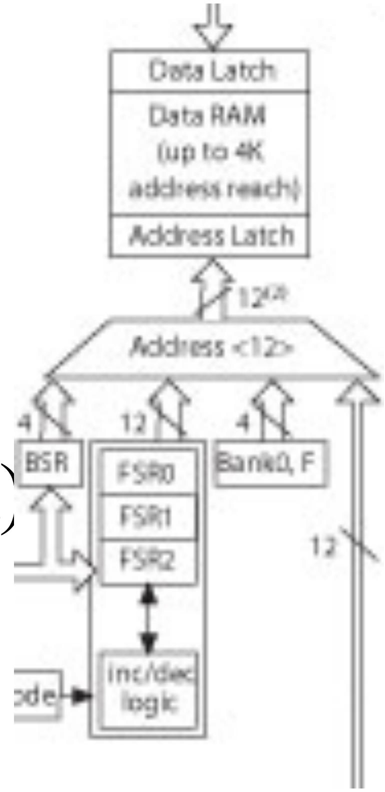
d-bit

d = 0 WREG
d = 1 data memory address

- d-bit refers to destination E.g., d=1, the result will go into data memory
- a-bit determines if we are accessing the access bank or BANK

Basic Programming Model

- Note that the RAM (file register or data memory) can be access via the following
  - BSR + 8-bit
  - FSR (three File Select Registers - FSR)
- When ACCESS BANK is selected
  - BANK0,F + 4-bits



Basic Programming Model

d-bit

d = 0 WREG

d = 1 data memory address

a-bit

a = 0 access bank

a = 1 use BSR

Examples:

Label	Op-code	Operand	Comment
Start:	MOVLW	0x00	;load WREG with 0x00
	GOTO	Start	;repeat

```
        MOVLW    0x06               ;place a 0x06 into W
        ADDLW    0x02               ;add a 0x02 to W
        MOVWF    0x00, 0            ;copy W to access bank register 0x00

;   OR  another version using the ACCESS keyword

        MOVLW    0x06               ;place a 0x06 into W
        ADDLW    0x02               ;add a 0x02 to W
        MOVWF    0x00, ACCESS       ;copy W to access bank register 0x00
```

Basic Programming Model

d-bit

d = 0   WREG

d = 1   data memory address

a-bit

a = 0 access bank

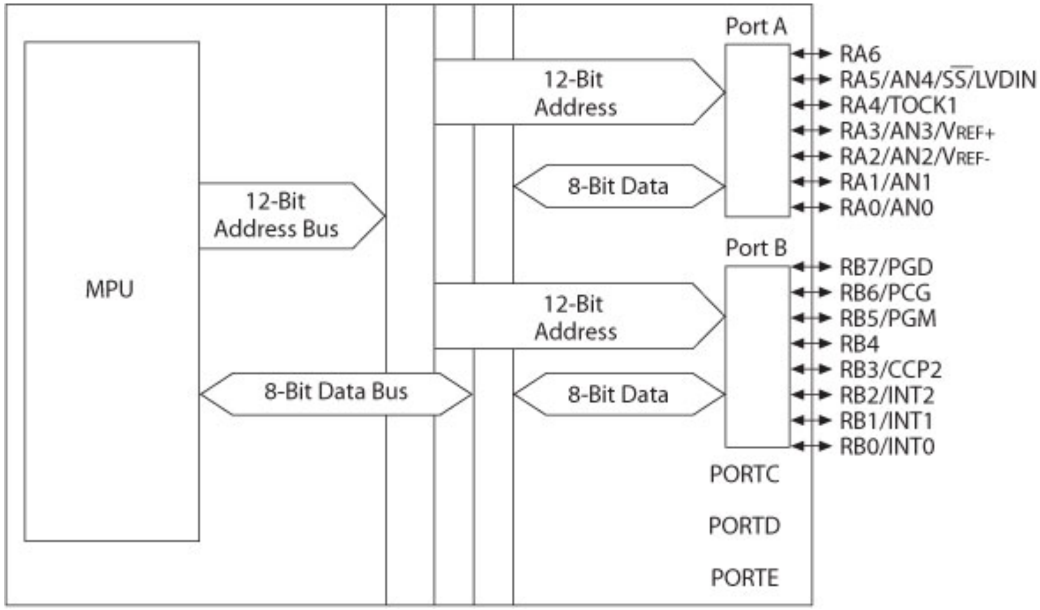
a = 1 use BSR

Examples:

```
          MOVLW    0x06                  ;place a 0x06 into W
          ADDLW    0x02                  ;add a 0x02 to W
          MOVLB    2                     ;load BSR with bank 2
          MOVWF    0x00, 1               ;copy W to data register 0x00
                                         ;of bank 2 or address 0x200

   ; OR using the BANKED keyword

          MOVLW 0x06                     ;place a 0x06 into W
          ADDLW    0x02                  ;add a 0x02 to W
          MOVLB    2                     ;load BSR with 2
          MOVLF    0x00, BANKED          ;copy W to data register 0x00
                                         ;of bank 2 or address 0x200

   ; OR without any bank indication

          MOVLW    0x06                  ;place a 0x06 into W
          ADDLW    0x02                  ;add a 0x02 to W
          MOVLB    2                     ;load BSR with bank 2
          MOVWF    0x00                  ;copy W to data register 0x00
                                         ;of bank 2 or address 0x200
```

PIC18F452 I/O Ports

- Five I/O ports
  - PORT A through PORT E
  - Most I/O pins are multiplexed
  - Generally have eight I/O pins with a few exceptions
  - Addresses already assigned to these ports in the design stage
  - Each port is identified by its assigned Special Function Registers (SFR) – look at the previous slide
    - PORTA (address of F80)
    - PORTB (address of F81)
    - → these are part of data memory or register file



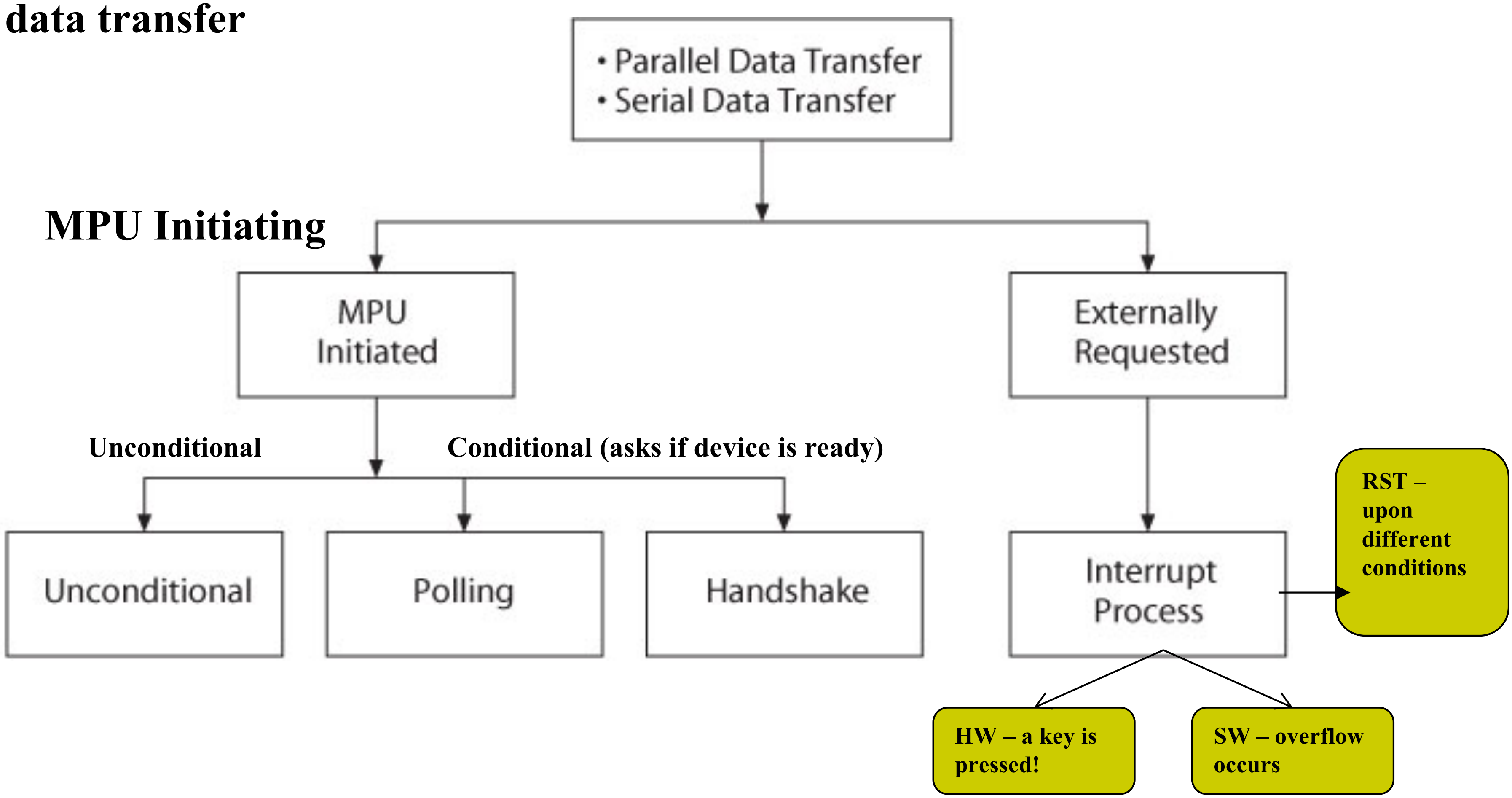
TRISB must be set to specify signal direction of PORT B.

Processes and Conditions of Data Transfer

- Interrupt is a process of communication between two devices
  - If provides efficient communication between the two devices
  - Examples: Sending a file to a printer, pressing a key on the key board
- External or Internal to the MPU

Processes and Conditions of Data Transfer

Parallel data transfer
Serial data transfer



- Parallel Data Transfer
- Serial Data Transfer

MPU Initiating

MPU Initiated

Externally Requested

Unconditional

Conditional (asks if device is ready)

Unconditional

Polling

Handshake

Interrupt Process

RST – upon different conditions

HW – a key is pressed!

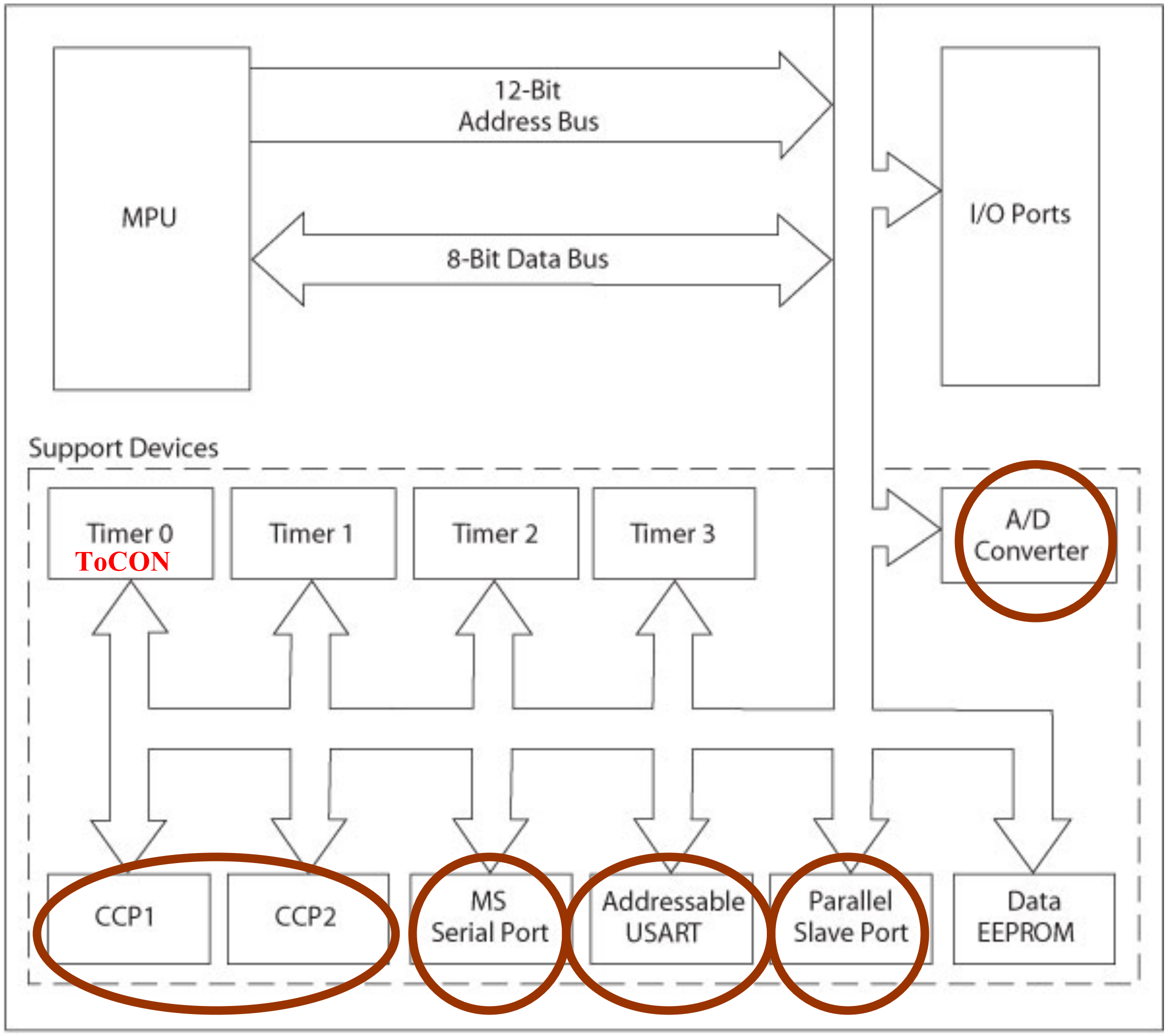
SW – overflow occurs

Processes and Conditions of Data Transfer

- Reset
  - Special type of external interrupt
  - Examples:
    - Manual Reset
    - Power-on Reset
    - Brown-out Reset (power goes below a specifies value)

MCU Support Devices (1 of 2)

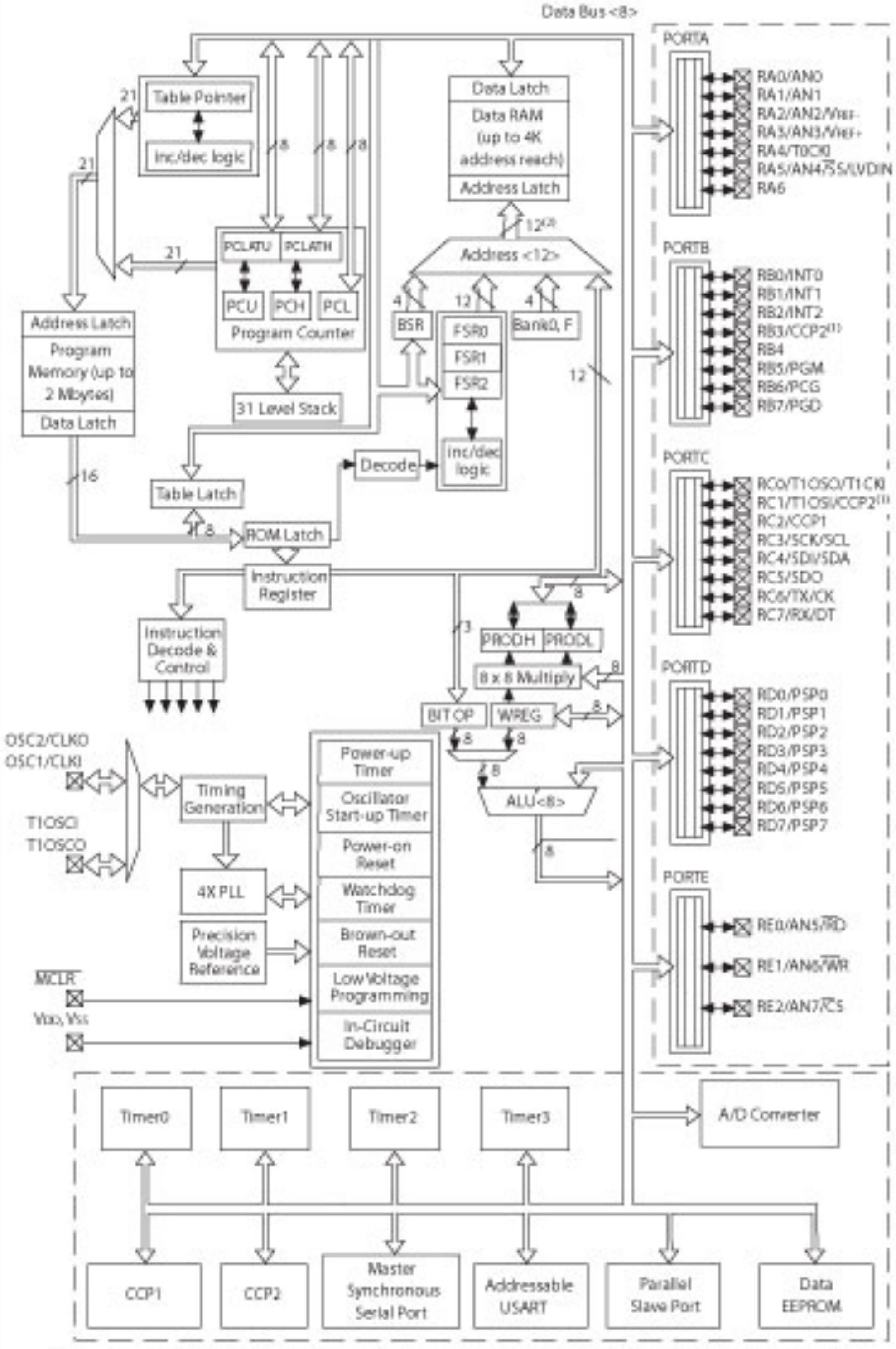
- Timers
  - A value is loaded in the register and continue changing at every clock cycle – time can be calculated
  - Can count on rising or falling edge
  - There are several timers: 8-bit, 16-bit
  - Controlled by SFR
- Master Synchronous Serial Port (MSSP)
  - Serial interface supporting RS232
- Addressable USART
  - Universal Sync/Async Rec/Transmitter
  - Another serial data communication
  - Similar to modem interfacing – also supports transfer between two microcontrollers to enhance IO ports
- A/D converter
  - 10-bit
  - Accepts analog signals from 13 channels
- Parallel Slave Port (PSP)
  - Used for interfacing with other MPU or MCU
- Capture, Compare and PWM (CCP Module)



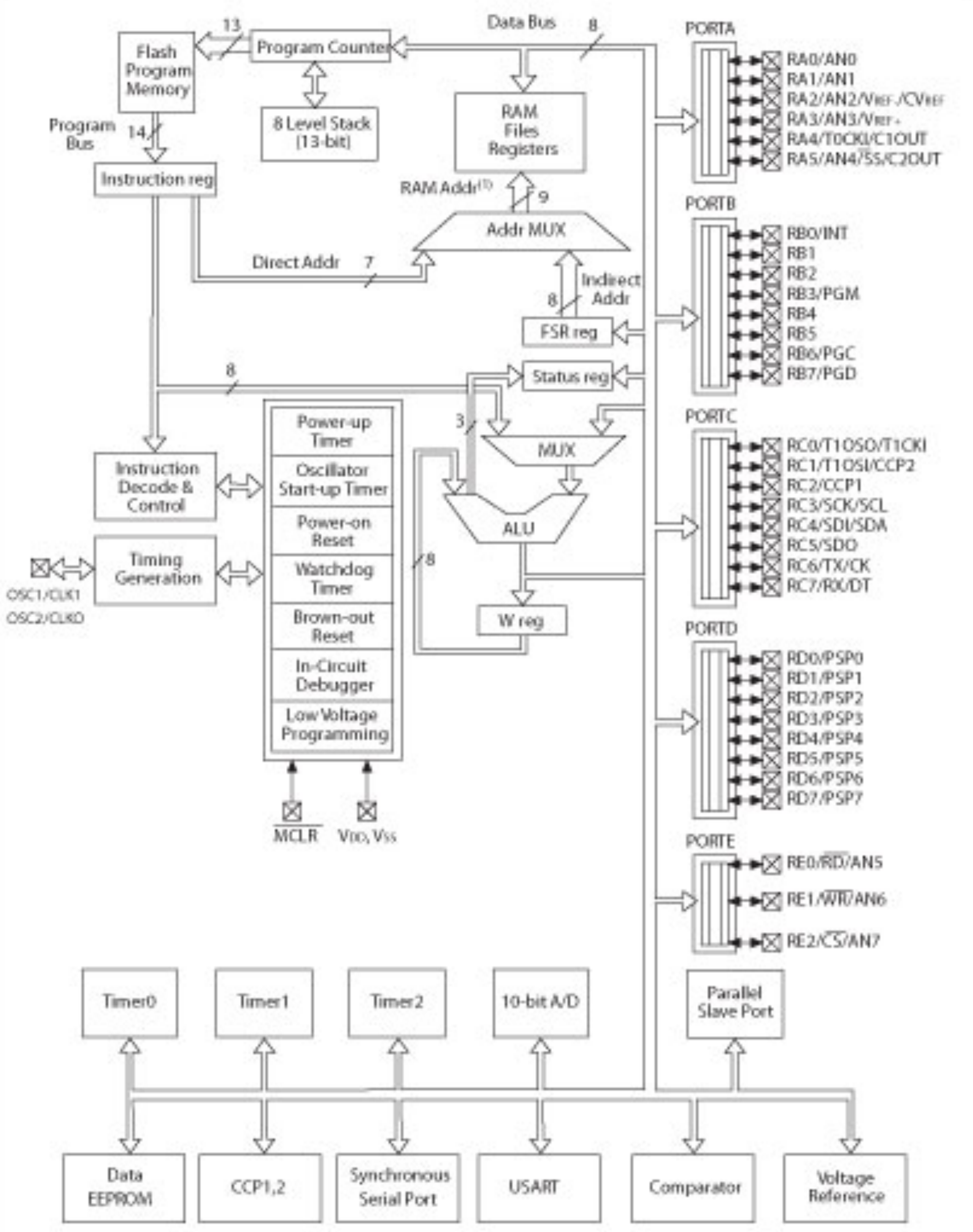
PIC18F Special Features

- Sleep mode
  - Power-down mode
- Watchdog timer (WDT)
  - Able to reset the processor if the program is caught in unknown state (e.g., infinite loop)
- Code protection
  - EEPROM can be protected through SFR
- In-circuit serial programming
- In-circuit debugger

PIC18F4X2 Architecture Block Diagram (page 46)



PIC16F87 Architecture Block Diagram



Device	Program Flash	Data Memory	Data EEPROM
PIC16F874A	4K words	192 Bytes	128 Bytes
PIC16F877A	8K words	368 Bytes	256 Bytes

Note 1: Higher order bits are from the Status register.

PIC18F Instructions and Assembly Language

- Has 77 instructions
  - Earlier PIC family of microcontrollers have either 33 or 35 instructions (Table 2-1)
- In PIC18F instruction set, all instructions are 16-bit word length except four instructions that are 32-bit length

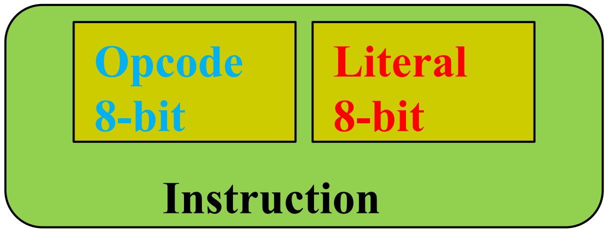
Instruction Description and Illustrations

- Copy (Load) 8-bit number into W register
  - Mnemonics: MOVLW 8-bit
  - Binary format:

0000 1110 **XXXX XXXX** (any 8-bit number)

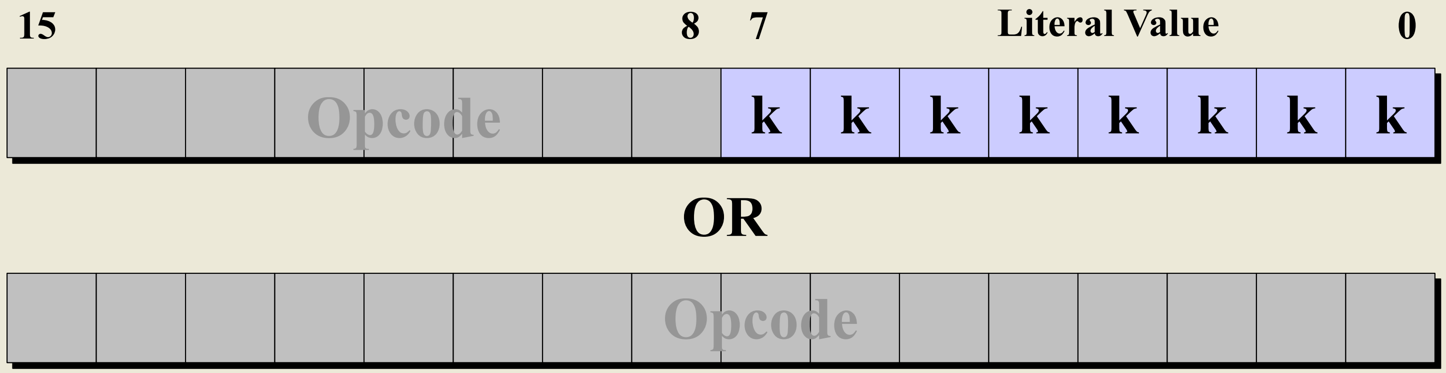
- Copy Contents of W register in PORTC
  - Mnemonics: MOVWF PORTC, a
    - (‘a’ indicates that PORTC is in the Access Bank)
  - Binary format:

0000 1110 1000 0010 (82H is PORTC address)



Instruction Set Overview

Literal and Control Operations



```
MOVLW   0x25
```

↑

Literal Value

Illustration: Displaying a Byte at an I/O Port (1 of 5)

- Problem statement:
  - Write instructions to light up alternate LEDs at PORTC.
- Hardware:
  - PORTC
    - bidirectional (input or output) port; should be setup as output port for display
  - Logic 1 will turn on an LED in Figure 2.10.

Illustration (2 of 5)

- Interfacing LEDs to PORTC
- Port C is F82H
- Note that PORT C is set to be an output!
- Hence, TRISC (address 94H) has to be set to 0

TRISC=0

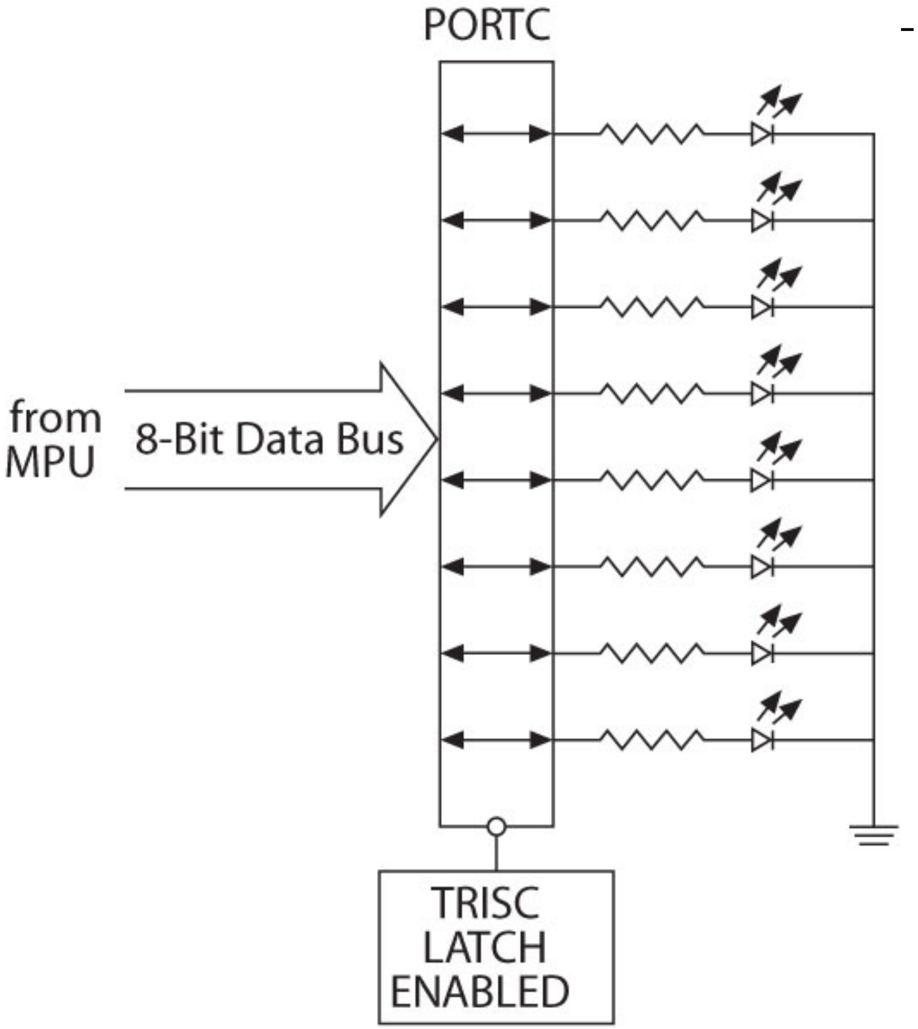
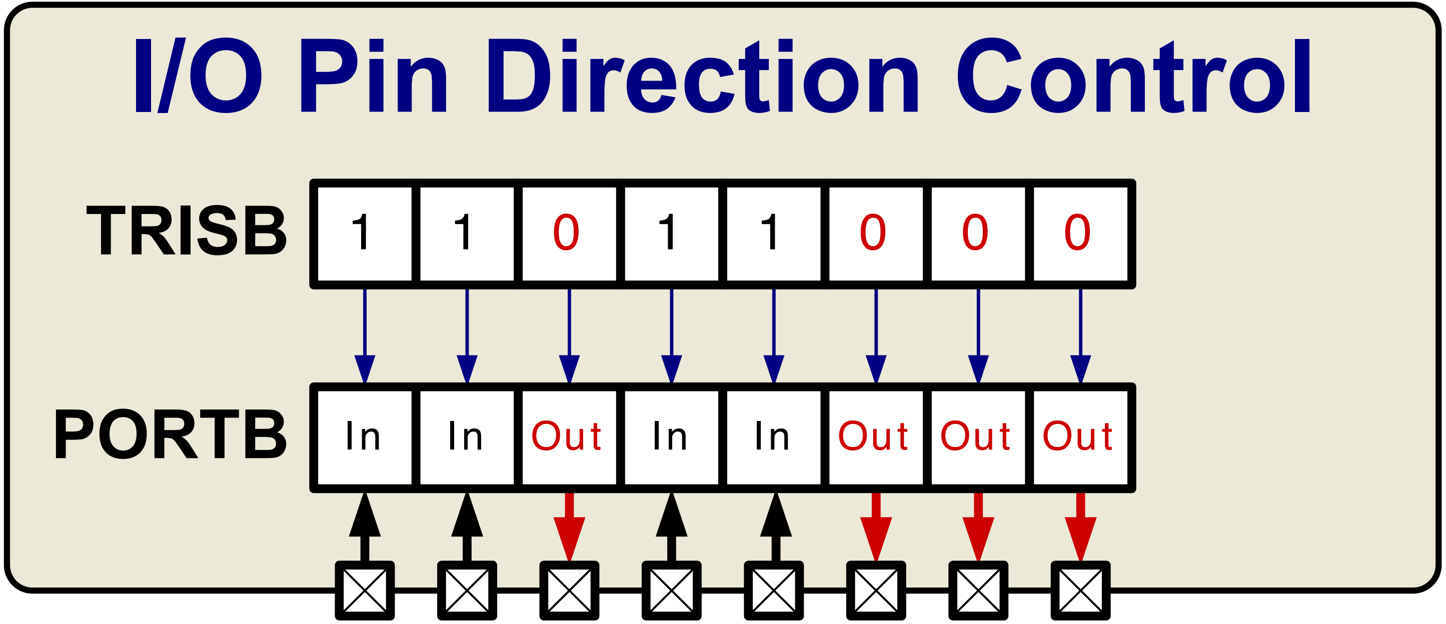


Illustration (3 of 5)

- Program (software)
  - Logic 0 to TRISC sets up PORTC as an output port
  - Byte 55H turns on alternate LEDs

    ```
    MOVLW     0x7F
    MOVWF     ADCON1          ;select all digital pins for ports
    MOVLW  00            ;Load W register with 0
    MOVWF TRISC, 0       ;Set up PORTC as output
    MOVLW 0x55           ;Byte 55H to turn on LEDS
    MOVWF PORTC,0        ;Turn on LEDs
    SLEEP                ;Power down
    ```

IO Port Access



I/O Pin Direction Control

TRISB	1	1	0	1	1	0	0	0
PORTB	In	In	Out	In	In	Out	Out	Out

- □ Bit n in TRISx controls the data direction of Bit n in PORTx
- □ 1 = Input, 0 = Output

Analog or Digital I/O?

- Some I/O pins multiplexed with analog inputs (analog by default)
- ADCON1 used to determine whether pin is analog in or digital I/O

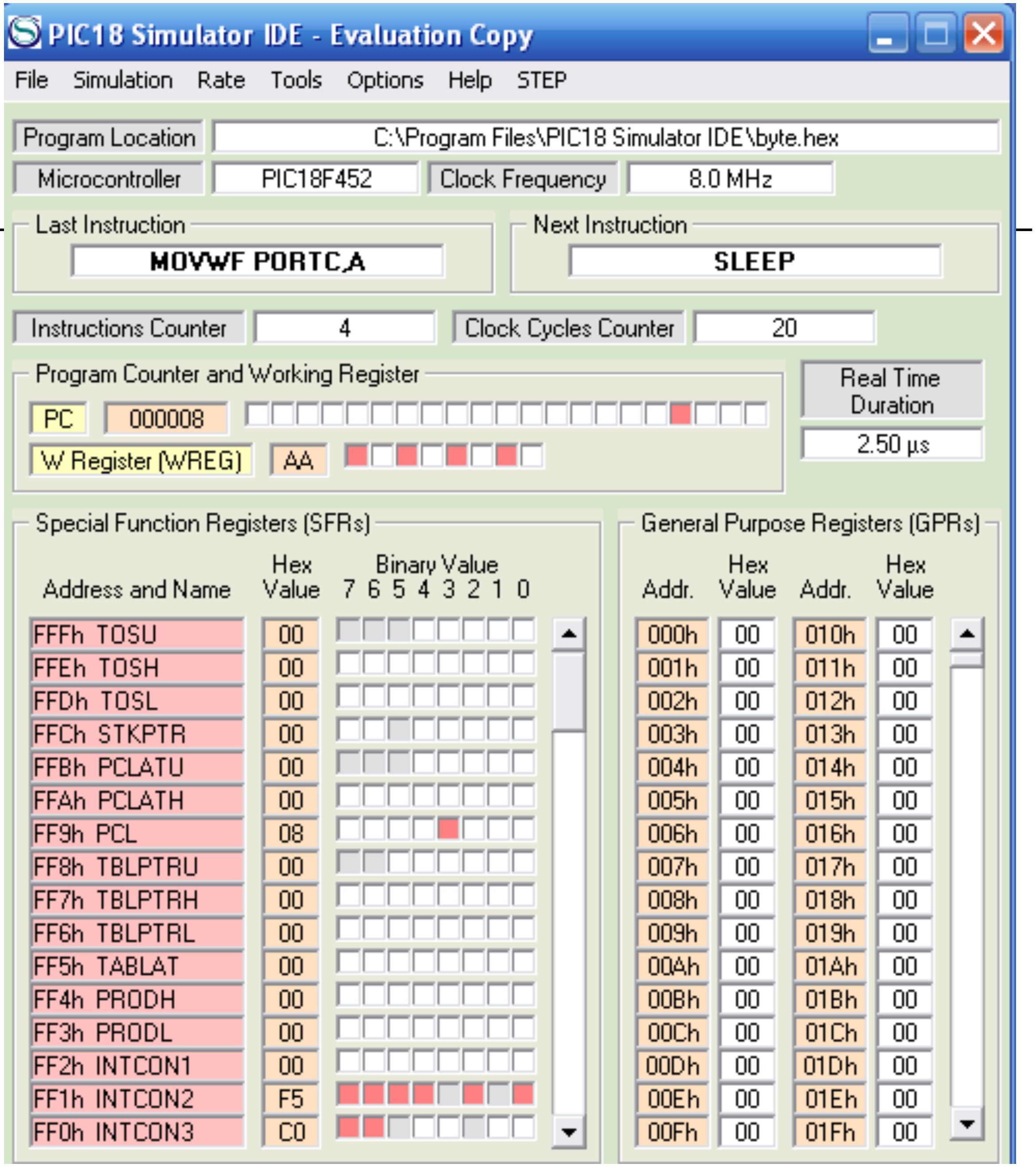
Bit 7							Bit 0
-	-	X	X	1	1	1	1
		VCFG1	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0

Set lower 4 bits to ‘1’ to make all multiplexed pins digital

PIC18 Simulator

- □ Using the Program Memory editor type in the opcode MOVLW 00 and MOWWF TRISC,0 as described in page 52 of your textbook.
- □ Run the program in step-by-step mode and observe the PC.
- □ Observe how the NEXT INSTRUCTION changes.
- □ What is the value of final clock cycle?
- □ How long does it take to complete the program in sec.?

- PIC18 Simulator IDE



PIC18 Simulator IDE - Evaluation Copy

Program Location: C:\Program Files\PIC18 Simulator IDE\byte.hex

Microcontroller: PIC18F452

Clock Frequency: 8.0 MHz

Last Instruction: **MOVWF PORTC,A**

Next Instruction: **SLEEP**

Instructions Counter: 4

Clock Cycles Counter: 20

Program Counter and Working Register

PC: 000008 □□□□□□□□□□□□□□□□□■□□□

W Register (WREG): AA ■□■□■□■□

Real Time Duration: 2.50 µs

Special Function Registers (SFRs)

Address and Name	Hex Value	7	6	5	4	3	2	1	0
FFFh TOSU	00	▨	▨	▨	□	□	□	□	□
FFEh TOSH	00	□	□	□	□	□	□	□	□
FFDh TOSL	00	□	□	□	□	□	□	□	□
FFCh STKPTR	00	□	□	▨	□	□	□	□	□
FFBh PCLATU	00	▨	▨	▨	□	□	□	□	□
FFAh PCLATH	00	□	□	□	□	□	□	□	□
FF9h PCL	08	□	□	□	□	■	□	□	□
FF8h TBLPTRU	00	▨	▨	□	□	□	□	□	□
FF7h TBLPTRH	00	□	□	□	□	□	□	□	□
FF6h TBLPTRL	00	□	□	□	□	□	□	□	□
FF5h TABLAT	00	□	□	□	□	□	□	□	□
FF4h PRODH	00	□	□	□	□	□	□	□	□
FF3h PRODL	00	□	□	□	□	□	□	□	□
FF2h INTCON1	00	□	□	□	□	□	□	□	□
FF1h INTCON2	F5	■	■	■	■	▨	■	▨	■
FF0h INTCON3	C0	■	■	▨	□	□	▨	□	□

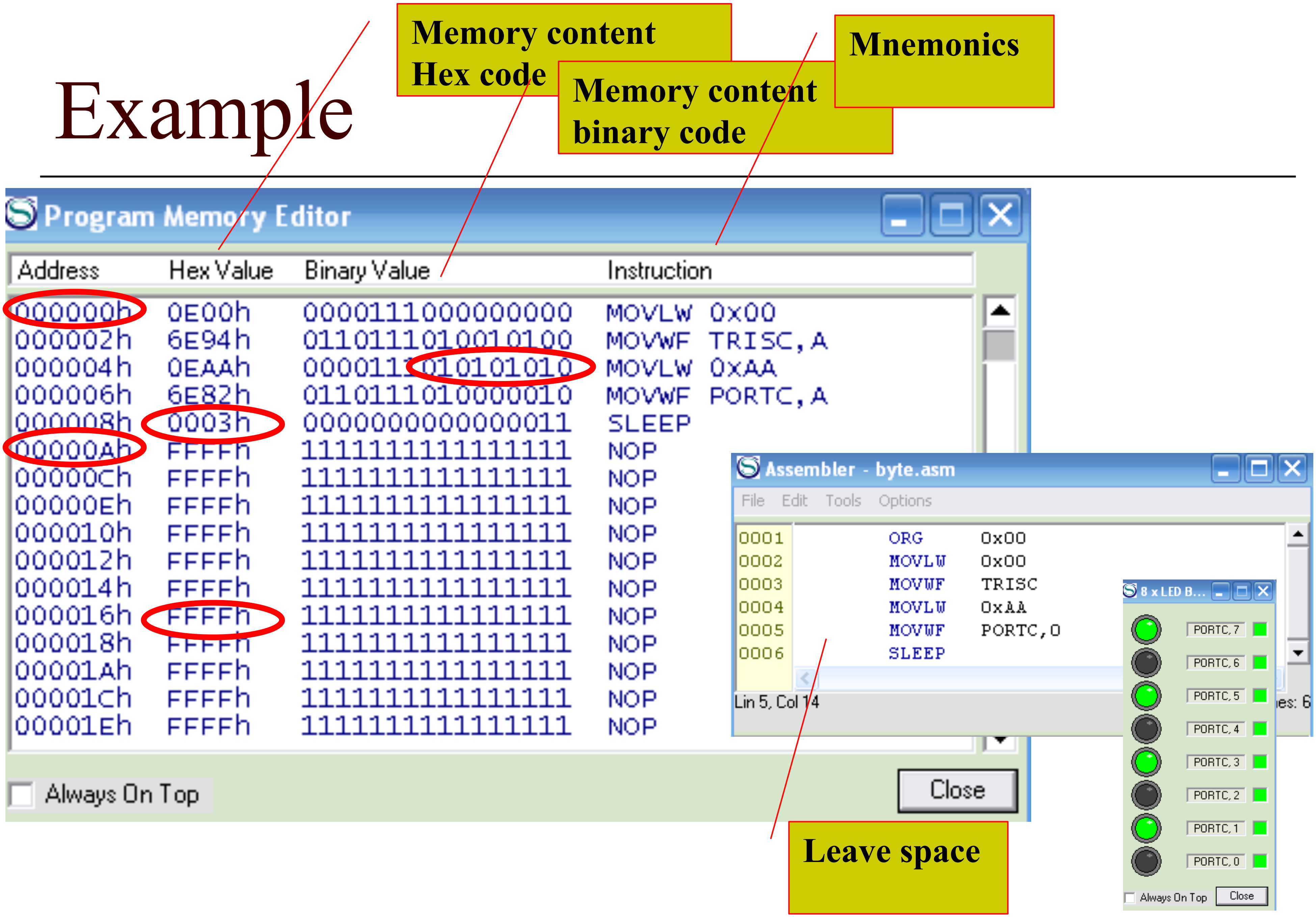
General Purpose Registers (GPRs)

Addr.	Hex Value	Addr.	Hex Value
000h	00	010h	00
001h	00	011h	00
002h	00	012h	00
003h	00	013h	00
004h	00	014h	00
005h	00	015h	00
006h	00	016h	00
007h	00	017h	00
008h	00	018h	00
009h	00	019h	00
00Ah	00	01Ah	00
00Bh	00	01Bh	00
00Ch	00	01Ch	00
00Dh	00	01Dh	00
00Eh	00	01Eh	00
00Fh	00	01Fh	00

Questions - PIC18 Simulator IDE

- What is the address for TRISC? SFR → F94
- What is the address for PORTE?
- How many SFR registers we have? FFF-F80
- How many GPR? 000-5FF
- How many bit PC has? 21

Example



Memory content Hex code

Memory content binary code

Mnemonics

Leave space

Program Memory Editor

Address	Hex Value	Binary Value	Instruction
000000h	0E00h	0000111000000000	MOVLW 0x00
000002h	6E94h	0110111010010100	MOVWF TRISC,A
000004h	0EAAh	0000111010101010	MOVLW 0XAA
000006h	6E82h	0110111010000010	MOVWF PORTC,A
000008h	0003h	0000000000000011	SLEEP
00000Ah	FFFFh	1111111111111111	NOP
00000Ch	FFFFh	1111111111111111	NOP
00000Eh	FFFFh	1111111111111111	NOP
000010h	FFFFh	1111111111111111	NOP
000012h	FFFFh	1111111111111111	NOP
000014h	FFFFh	1111111111111111	NOP
000016h	FFFFh	1111111111111111	NOP
000018h	FFFFh	1111111111111111	NOP
00001Ah	FFFFh	1111111111111111	NOP
00001Ch	FFFFh	1111111111111111	NOP
00001Eh	FFFFh	1111111111111111	NOP

☐ Always On Top

Assembler - byte.asm

```
        ORG     0x00
        MOVLW   0x00
        MOVWF   TRISC
        MOVLW   0xAA
        MOVWF   PORTC,0
        SLEEP
```

Lin 5, Col 14

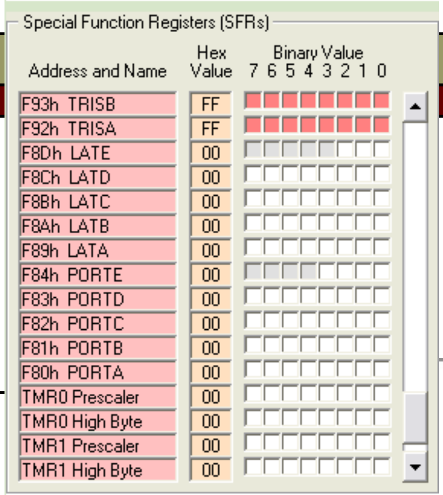
Illustration (4 of 5)

- □ Execution of the instruction:

WREG=AA

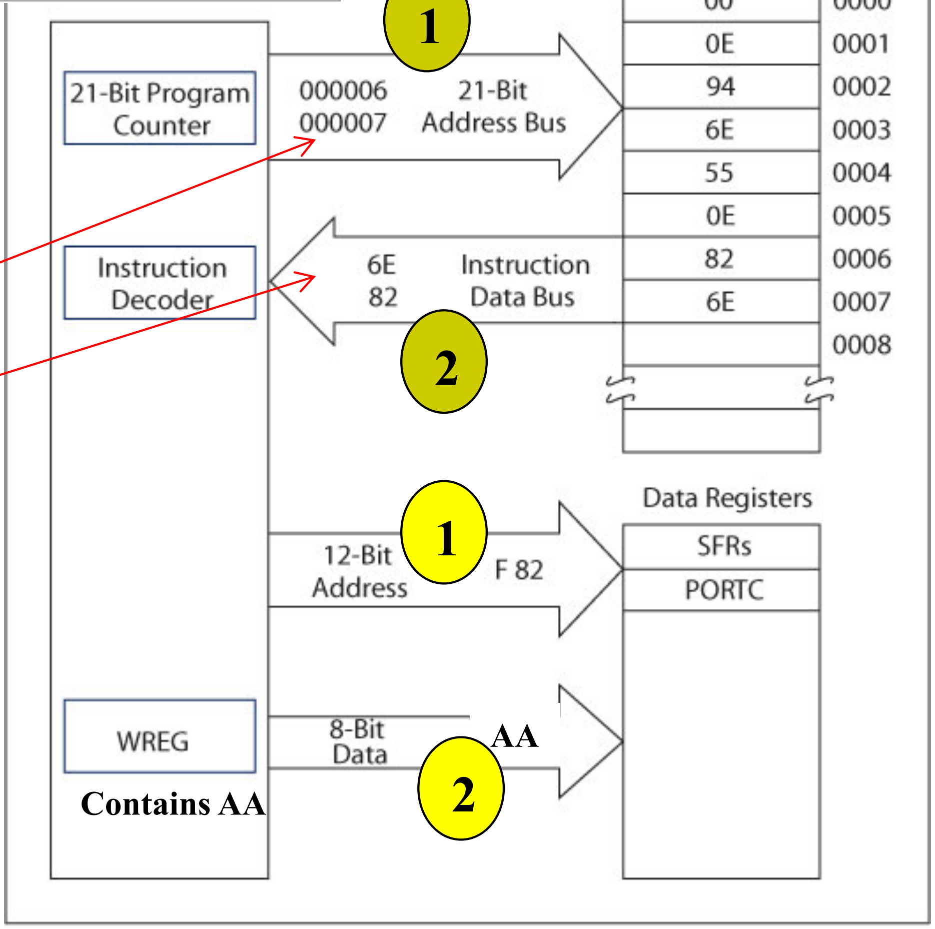
MOVWF PORTC

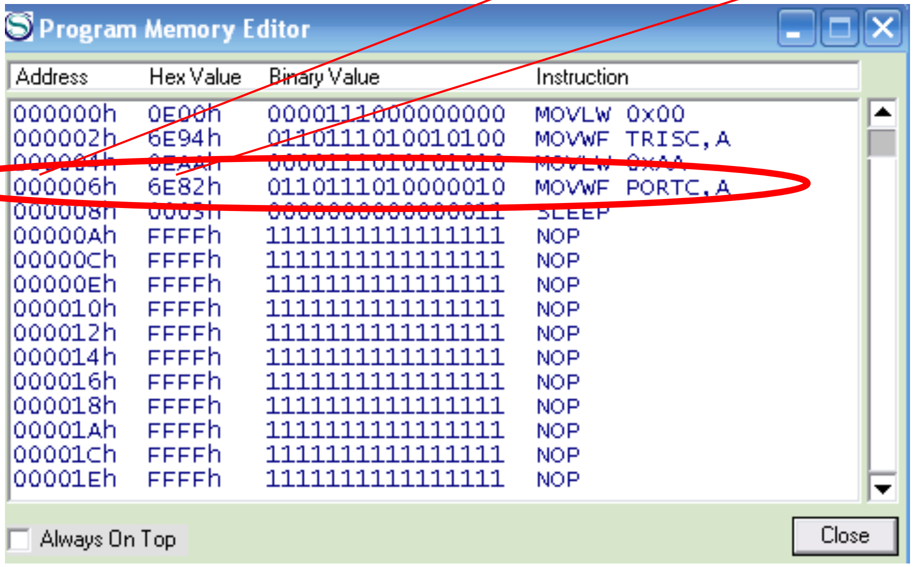
Copy from WREG→PORT C (82_H)



Special Function Registers (SFRs)

Address and Name	Hex Value	Binary Value 7 6 5 4 3 2 1 0
F93h TRISB	FF	🟥🟥🟥🟥🟥🟥🟥🟥
F92h TRISA	FF	🟥🟥🟥🟥🟥🟥🟥🟥
F8Dh LATE	00	□□□□□□□□
F8Ch LATD	00	□□□□□□□□
F8Bh LATC	00	□□□□□□□□
F8Ah LATB	00	□□□□□□□□
F89h LATA	00	□□□□□□□□
F84h PORTE	00	□□□□□□□□
F83h PORTD	00	□□□□□□□□
F82h PORTC	00	□□□□□□□□
F81h PORTB	00	□□□□□□□□
F80h PORTA	00	□□□□□□□□
TMR0 Prescaler	00	□□□□□□□□
TMR0 High Byte	00	□□□□□□□□
TMR1 Prescaler	00	□□□□□□□□
TMR1 High Byte	00	□□□□□□□□

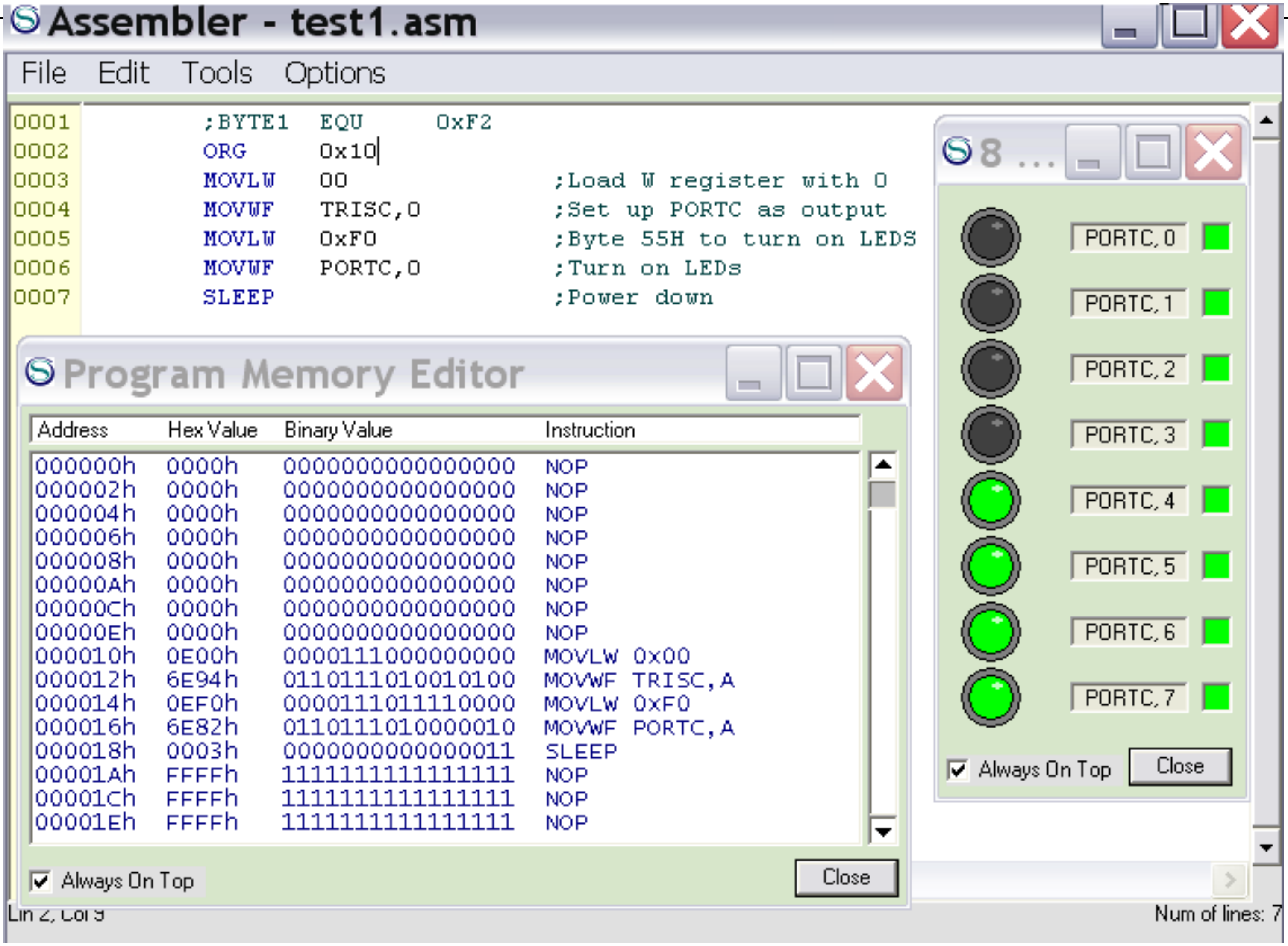




Program Memory Editor

```
Address   Hex Value  Binary Value       Instruction
000000h   0E00h      0000111000000000   MOVLW  0x00
000002h   6E94h      0110111010010100   MOVWF  TRISC,A
000004h   0EAAh      0000111010101010   MOVLW  0xAA
000006h   6E82h      0110111010000010   MOVWF  PORTC,A
000008h   0003h      0000000000000011   SLEEP
00000Ah   FFFFh      1111111111111111   NOP
00000Ch   FFFFh      1111111111111111   NOP
00000Eh   FFFFh      1111111111111111   NOP
000010h   FFFFh      1111111111111111   NOP
000012h   FFFFh      1111111111111111   NOP
000014h   FFFFh      1111111111111111   NOP
000016h   FFFFh      1111111111111111   NOP
000018h   FFFFh      1111111111111111   NOP
00001Ah   FFFFh      1111111111111111   NOP
00001Ch   FFFFh      1111111111111111   NOP
00001Eh   FFFFh      1111111111111111   NOP
```

Another Example



```
        ;BYTE1  EQU     0xF2
        ORG     0x10
        MOVLW   00                  ;Load W register with 0
        MOVWF   TRISC,0             ;Set up PORTC as output
        MOVLW   0xF0                ;Byte 55H to turn on LEDS
        MOVWF   PORTC,0             ;Turn on LEDs
        SLEEP                       ;Power down
```

```
Address   Hex Value  Binary Value        Instruction
000000h   0000h      0000000000000000    NOP
000002h   0000h      0000000000000000    NOP
000004h   0000h      0000000000000000    NOP
000006h   0000h      0000000000000000    NOP
000008h   0000h      0000000000000000    NOP
00000Ah   0000h      0000000000000000    NOP
00000Ch   0000h      0000000000000000    NOP
00000Eh   0000h      0000000000000000    NOP
000010h   0E00h      0000111000000000    MOVLW  0x00
000012h   6E94h      0110111010010100    MOVWF  TRISC,A
000014h   0EF0h      0000111011110000    MOVLW  0xF0
000016h   6E82h      0110111010000010    MOVWF  PORTC,A
000018h   0003h      0000000000000011    SLEEP
00001Ah   FFFFh      1111111111111111    NOP
00001Ch   FFFFh      1111111111111111    NOP
00001Eh   FFFFh      1111111111111111    NOP
```

References

- Good exercises: http://www.gooligum.com.au/tutorials.html
- Read the Wiki on Microchip: http://en.wikipedia.org/wiki/PIC_microcontroller
- Flag simulator: http://www.ee.unb.ca/cgi-bin/tervo/alu.pl
- PIC Tutorial (flash-based speaking instructor will be tutoring you…): http://www.pictutorials.com/Flash_Tutorials.htm