

MSP430 Architecture

Today

- Learn about the MSP430 processor
- Lecture derived from:

MSP430FR57xx Family User's Guide

- Memory
- General Purpose Registers
- Control Registers
- Addressing Modes
- Memory Map

Object

- **Fundamentally the lowest element is a Bit.**
 - *A bit is a single location*
 - *This location has two possible values*
 - *Can contain either a “1” or “0”*

Object

- **The next element is a Nibble.**
 - *4 Bits together creates a Nibble*
 - *Can contain one of 16 possible values*

0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111

Object

**Much Easier to refer to the
binary Nibble as a Hex Value.**

Binary	Hex
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

Object

- **The next element is a Byte.**
 - *8 Bits together creates a Byte*
 - *2 Nibbles make a byte*
 - *This location has 256 possible values*
 - *Uses Hex notation to represent values*
 - *Usually contains identifiers to signify Hex*
 - *0x00*
 - *00h*
 - *Basic Reference for all larger values*
 - *Word [16 bit]*
 - *Long Word [32 bit]*
 - *Long Long Word [64 bit]*

Binary	Hex
00000000	0x00
00000001	0x01
00000010	0x02
00000011	0x03
00000100	0x04
00000101	0x05
00000110	0x06
00000111	0x07
00001000	0x08
00001001	0x09
00001010	0x0A
00001011	0x0B
00001100	0x0C
00001101	0x0D
00001110	0x0E
00001111	0x0F

Memory

- ***All memory is Byte size!***

- All Memory address are for a Byte location
- A byte contains 1 address.
- A word [2 bytes] contains 2 address
 - The lowest Address will return the full word.
- A long word [4 bytes] contains 4 address
 - The lowest Address will return the full long word.
- A long long word [8 bytes] contains 8 address
 - The lowest Address will return the full long long word.

Simple Memory Organization

- $k * m$ array of stored bits (k is usually 2^n)

- Location [Address]
 - unique (n -bit) identifier of location

- Contents [Data (8 bit)]
 - m -bit value stored in location

0000	
0001	
0010	
0011	00101101
0100	
0101	
0110	
	⋮
1101	10100010
1110	
1111	

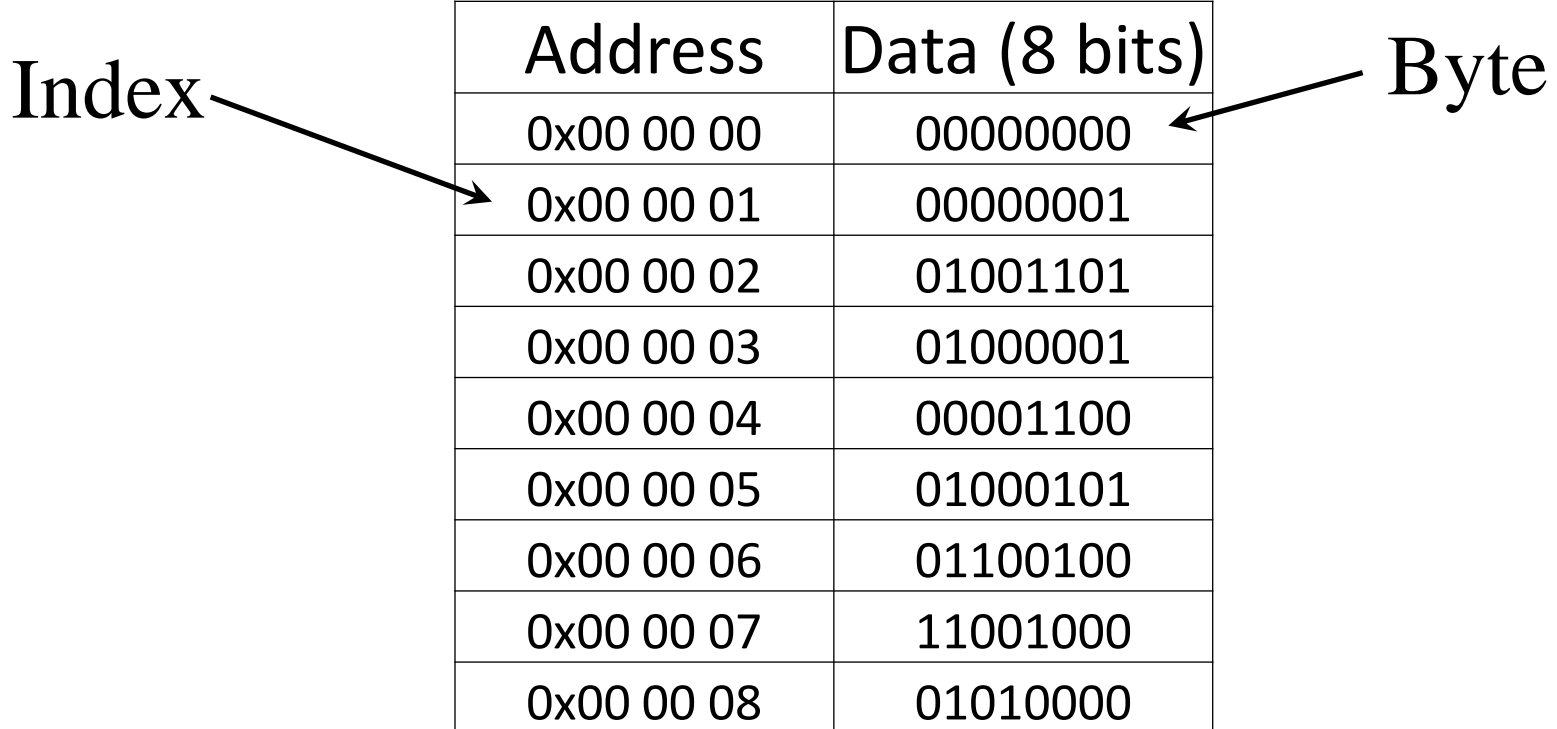
Simple Memory Organization

•Contents [Data (8 *bit*)]

Address	Data	
	Binary	Hex
0x00 00 00	00000000	0x00
0x00 00 01	00000001	0x01
0x00 00 02	01001101	0x4D
0x00 00 03	01000001	0x41
0x00 00 04	00001100	0x0C
0x00 00 05	01000101	0x45
0x00 00 06	01100100	0x64
0x00 00 07	11001000	0xC8
0x00 00 08	01010000	0x50
0x00 00 09	00111001	0x39
0x00 00 0A	00001101	0x0D
0x00 00 0B	11011100	0xDC
0x00 00 0C	10101100	0xAC
0x00 00 0D	01100011	0x63
0x00 00 0E	01000111	0x47
0x00 00 0F	00110010	0x32

Simple Memory Organization

- Viewed as a large, single-dimensional array
- A memory address is an index into the array
- "Byte addressing" means that the index is the location for a byte of memory



The diagram illustrates memory organization. A table with two columns, 'Address' and 'Data (8 bits)', is shown. An arrow labeled 'Index' points to the first row's address '0x00 00 01'. Another arrow labeled 'Byte' points to the first row's data '00000000'.

Address	Data (8 bits)
0x00 00 00	00000000
0x00 00 01	00000001
0x00 00 02	01001101
0x00 00 03	01000001
0x00 00 04	00001100
0x00 00 05	01000101
0x00 00 06	01100100
0x00 00 07	11001000
0x00 00 08	01010000

Memory Organization

- Bytes are nice, but most data items use larger "words"
 - For MSP430, a word is 16 bits or 2 bytes.

Address	Data	
	Binary	Hex
0x00 00 00	00000000	0x00
0x00 00 01	00000001	0x01
0x00 00 02	01001101	0x4D
0x00 00 03	01000001	0x41
0x00 00 04	00001100	0x0C
0x00 00 05	01000101	0x45
0x00 00 06	01100100	0x64
0x00 00 07	11001000	0xC8

Address	Data			
	Binary		Hex	
0x00 00 00	00000001	00000000	0x01	0x00
0x00 00 02	01000001	01001101	0x41	0x4D
0x00 00 04	01000101	00001100	0x45	0x0C
0x00 00 06	11001000	01100100	0xC8	0x64

- 2^{16} bytes with byte addresses from 0, 1, 2 to $2^{16}-1$
- 2^{15} words with byte addresses 0, 2, 4, ... $2^{16}-2$

All memory locations are 8 bits

Endianness

- Big endian: most significant byte is stored at the lowest byte address
- Little endian: least significant byte is stored at the lowest address

◆ Ex: 68000, PowerPC, Sun SPARC

◆ Ex: x86, DEC VAX, Alpha

0	12
1	34
2	56
3	78
4	AB
5	CD
6	EF
7	01

BOTH store same data:
word 12345678 is at
location 0,
word ABCDEF01 is at
location 4

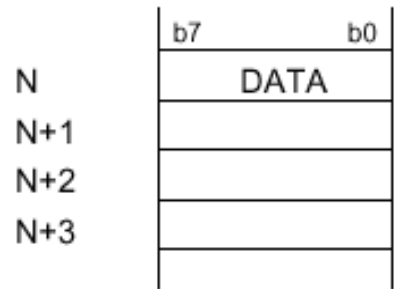
0	78
1	56
2	34
3	12
4	01
5	EF
6	CD
7	AB

- Most of the time we will avoid this issue in class by only loading/storing words or loading/storing bytes
- If two processors with different conventions use a local area network, a disk drive, etc., YOU need to pay attention to endianness

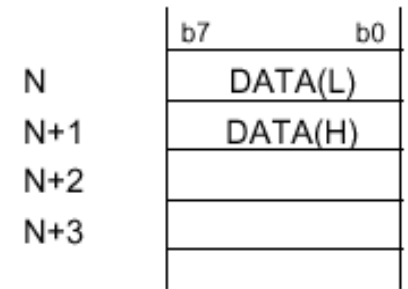
Data Formats for the MSP430

- Byte
 - 8 bits
 - signed & unsigned
 - .B suffix for instruction
- Word
 - 16 bits
 - signed & unsigned
 - .W suffix
- Address & long word
 - Limited to specific instructions

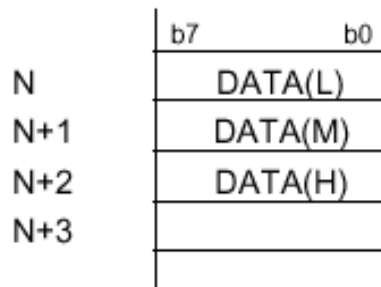
Is the MSP430 big or little endian?



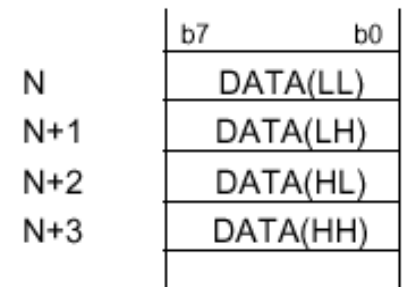
Byte (8-bit) data



Word (16-bit) data



20-bit (Address) data



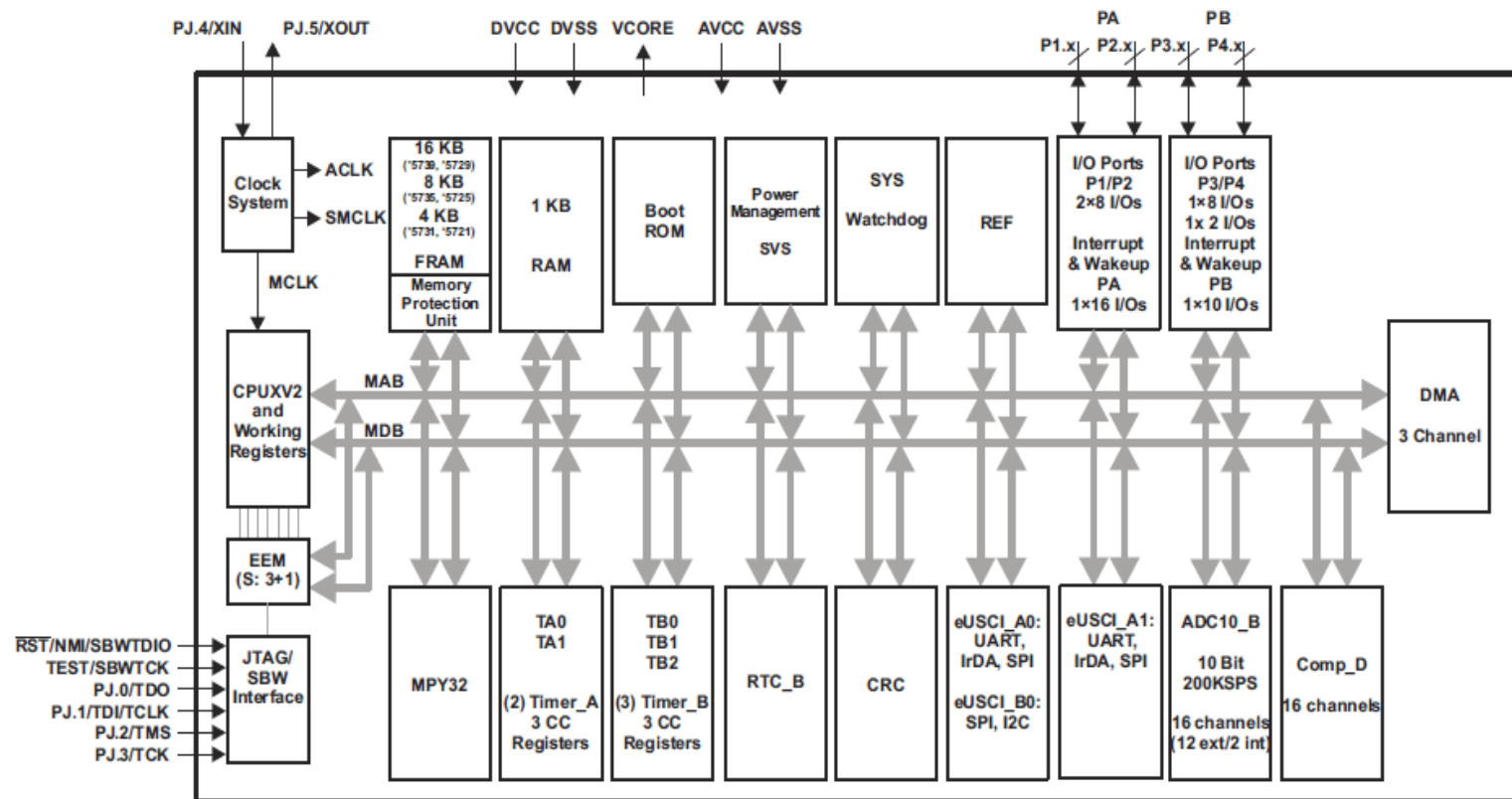
Long Word (32-bit) data

Review of the MSP430 MCU

Microcontroller has:

- General Purpose Registers, RAM, FRAM, Digital Ports, Analog Ports, Timers, Clocks, DMA Controller, RTC, Power Management, Watchdog, Multiply 32bit, CRC, Communication

Functional Block Diagram – MSP430FR5739IRHA



General Memory Map

Address		Size	Description	Type
Begin	End			
0x000000	0x000FFF	4 KB	Peripherals	Registers
0X001000	0X0011FF	512 B	Boot Strap Loader 0	ROM
0X001200	0X0013FF	512 B	Boot Strap Loader 1	ROM
0X001400	0X0015FF	512 B	Boot Strap Loader 2	ROM
0X001600	0X0017FF	512 B	Boot Strap Loader 3	ROM
0X001800	0X00187F	128 B	Info B	FRAM
0X001880	0X0018FF	128 B	Info A	FRAM
0X001900	0X00197F	128 B	Mirrored to Info B	FRAM
0X001980	0X0019FF	128 B	Mirrored to Info A	FRAM
0X001A00	0X001A7F	128 B	Device Descriptor Info	TLV FRAM
0X001C00	0X001FFF	1 KB	RAM	RAM
0X002000	0X00C1FF	40 KB	Not Used	
0X00C200	0x00FF7F	15 KB	Main: Code Memory	FRAM
0x00FF80	0x00FFFF	128 B	Main: Interrupt Vectors	FRAM

General Memory Map

Address		Size	Description	Type
Begin	End			
0x000000	0x000FFF	4 KB	Peripherals	Registers
0X001000	0X0011FF	512 B	Boot Strap Loader 0	ROM
0X001200	0X0013FF	512 B	Boot Strap Loader 1	ROM
0X001400	0X0015FF	512 B	Boot Strap Loader 2	ROM
0X001600	0X0017FF	512 B	Boot Strap Loader 3	ROM
0X001800	0X00187F	128 B	Info B	FRAM
0X001880	0X0018FF	128 B	Info A	FRAM
0X001900	0X00197F	128 B	Mirrored to Info B	FRAM
0X001980	0X0019FF	128 B	Mirrored to Info A	FRAM
0X001A00	0X001A7F	128 B	Device Descriptor Info	TLV FRAM
0X001C00	0X001FFF	1 KB	RAM	RAM
0X002000	0X00C1FF	40 KB	Not Used	
0X00C200	0x00FF7F	15 KB	Main: Code Memory	FRAM
0x00FF80	0x00FFFF	128 B	Main: Interrupt Vectors	FRAM

MSP430 Registers

- The MSP430 CPU has a 16-bit RISC architecture that is highly transparent to the application. All operations, other than program-flow instructions, are performed as register operations in conjunction with seven addressing modes for source operand and four addressing modes for destination operand.
- The CPU is integrated with **16 registers** that provide reduced instruction execution time. The register-to-register operation execution time is one cycle of the CPU clock.
- **Four of the registers, R0 to R3**, are dedicated as **program counter, stack pointer, status register, and constant generator**, respectively. The remaining registers are general-purpose registers.
- Peripherals are connected to the CPU using data, address, and control buses, and can be handled with all instructions.
- The instruction set consists of the original 51 instructions with three formats and seven address modes and additional instructions for the expanded address range.
- Each instruction can operate on word and byte data.

Use of Registers

- SP: Stack Pointer – for accessing call stack
 - USP: User code
 - ISP: Interrupt code
- FB: Frame Base – for accessing frame on call stack
- SB: Static Base
- INTB: Interrupt table pointer