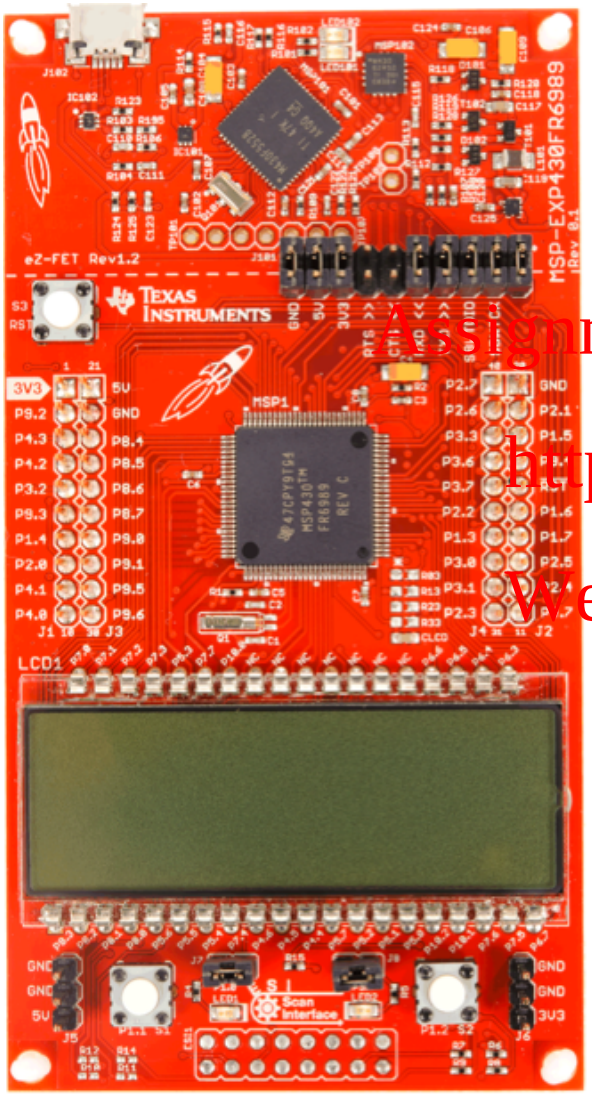


Lecture 5

Memory Map of MSP430FR6989

Coming Up Next

Friday we will start coding Finally!!

For that you will need your Launchpad and Code Composer Studio (CCS)

Posted a guide on how to install CCS – Find it under Resources in Carmen and follow the steps carefully

CCS Installation Guide.pdf

Download CCS Installation Guide.pdf (258 KB)

Last Time: Units of Memory

Pop Quiz: What is a kilobyte?

Is it 1kB = 1000 B?

Is it 1kB = 1024 B?

The SI unit prefix k (kilo) is always 1000 !

M (mega) is always 10^6 !

Powers of 10

But, there is a reason for measuring memory in powers of 2

Hence the prefixes for binary multiples: "**kilo binary**" or **kibi** written as **Ki**

1 KiB = 1024 B = 2^{10} B

1 MiB = 1,048,576 B = 2^{10} B

…

e.g., see https://physics.nist.gov/cuu/Units/binary.html

Last Time: Essential Components of an MCU

A **microcontroller** contains at the bare minimum

- Central processing unit (CPU)
- Program memory – **nonvolatile**
- Data memory – usually volatile
- Clock
- Address and data busses
- Input and output (I/O) ports

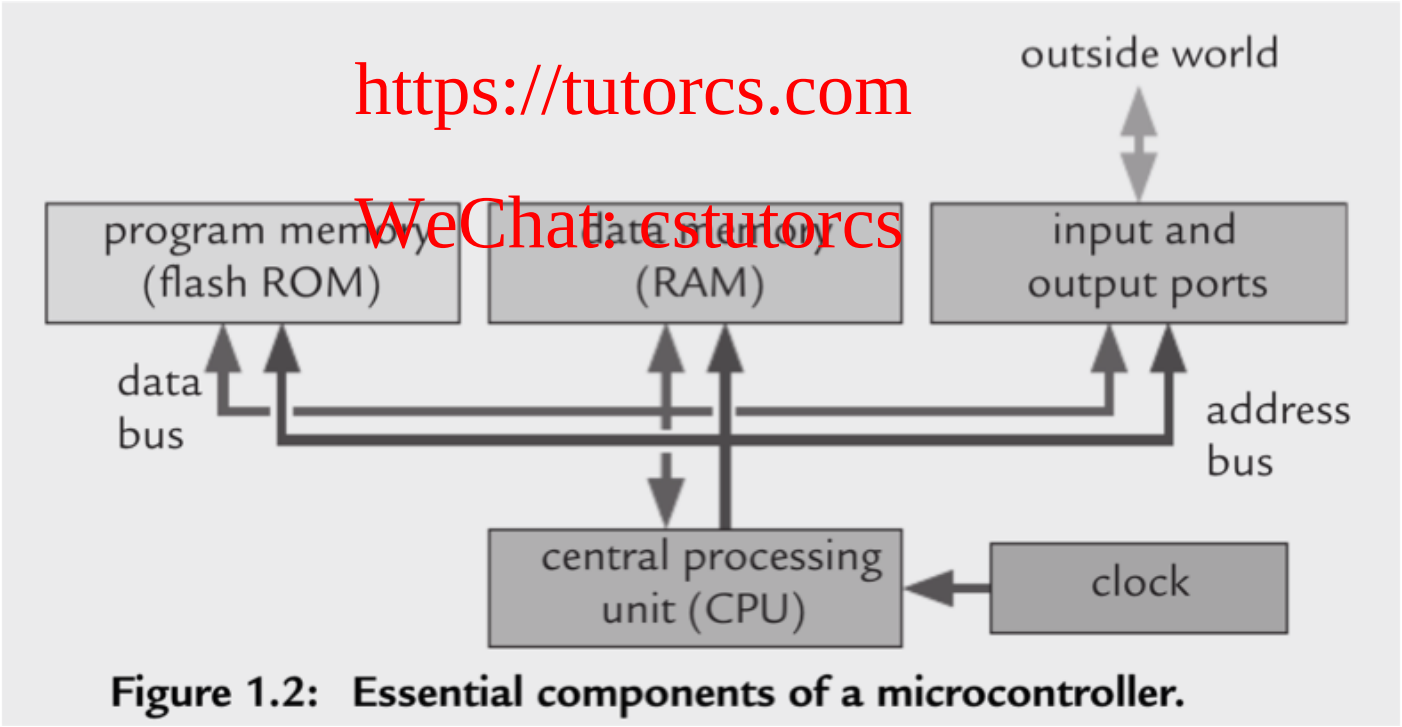
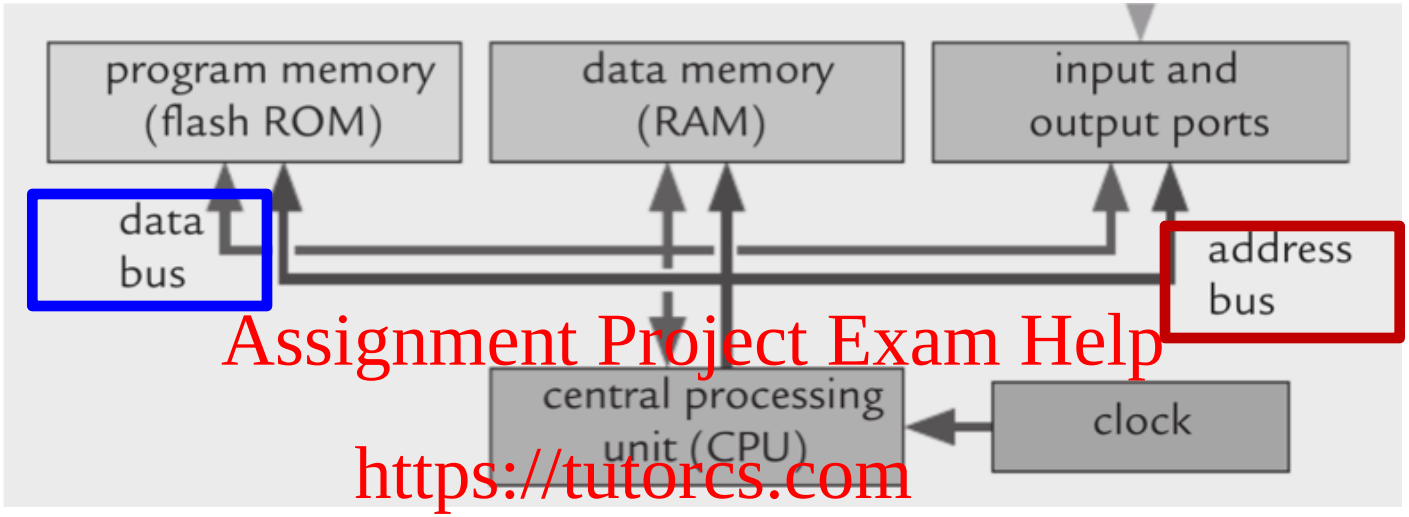


Figure 1.2: Essential components of a microcontroller.

Last Time: Essential Components of an MCU



All memory is linked to the CPU by busses for data, address and control

The width of the **data bus** determines the architecture of the MCU

e.g., 16-bit processor

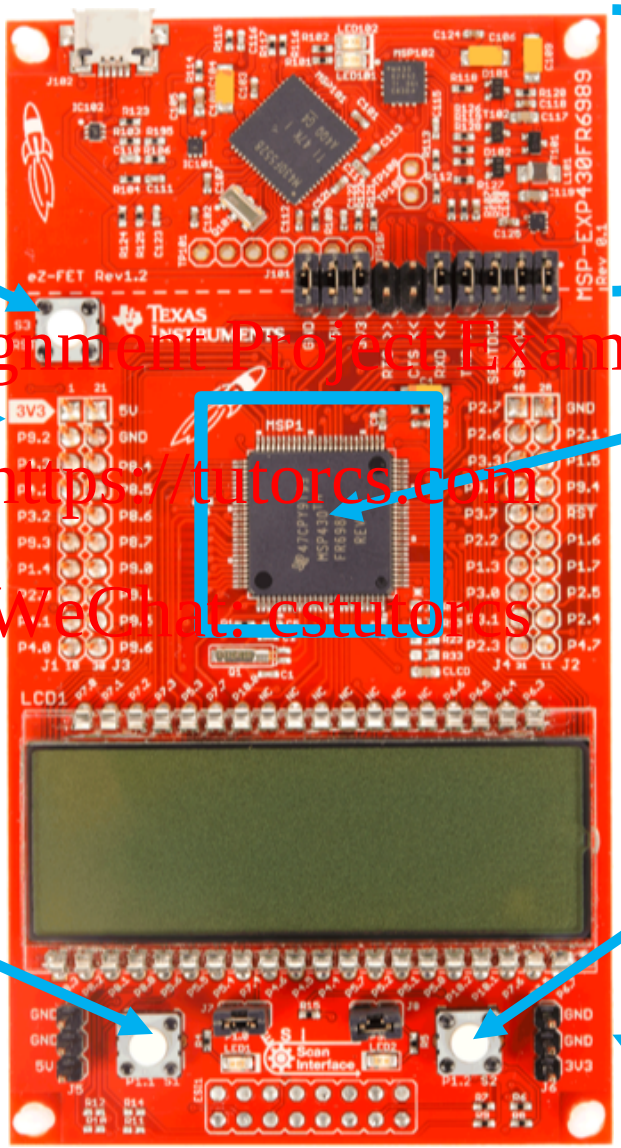
The width of the **address bus** determines the size of the memory that can be addressed

e.g., 16 bits can address 2^{16} = 65,536 different memory locations in **total**

i.e., program **and** data memory **and** peripheral registers

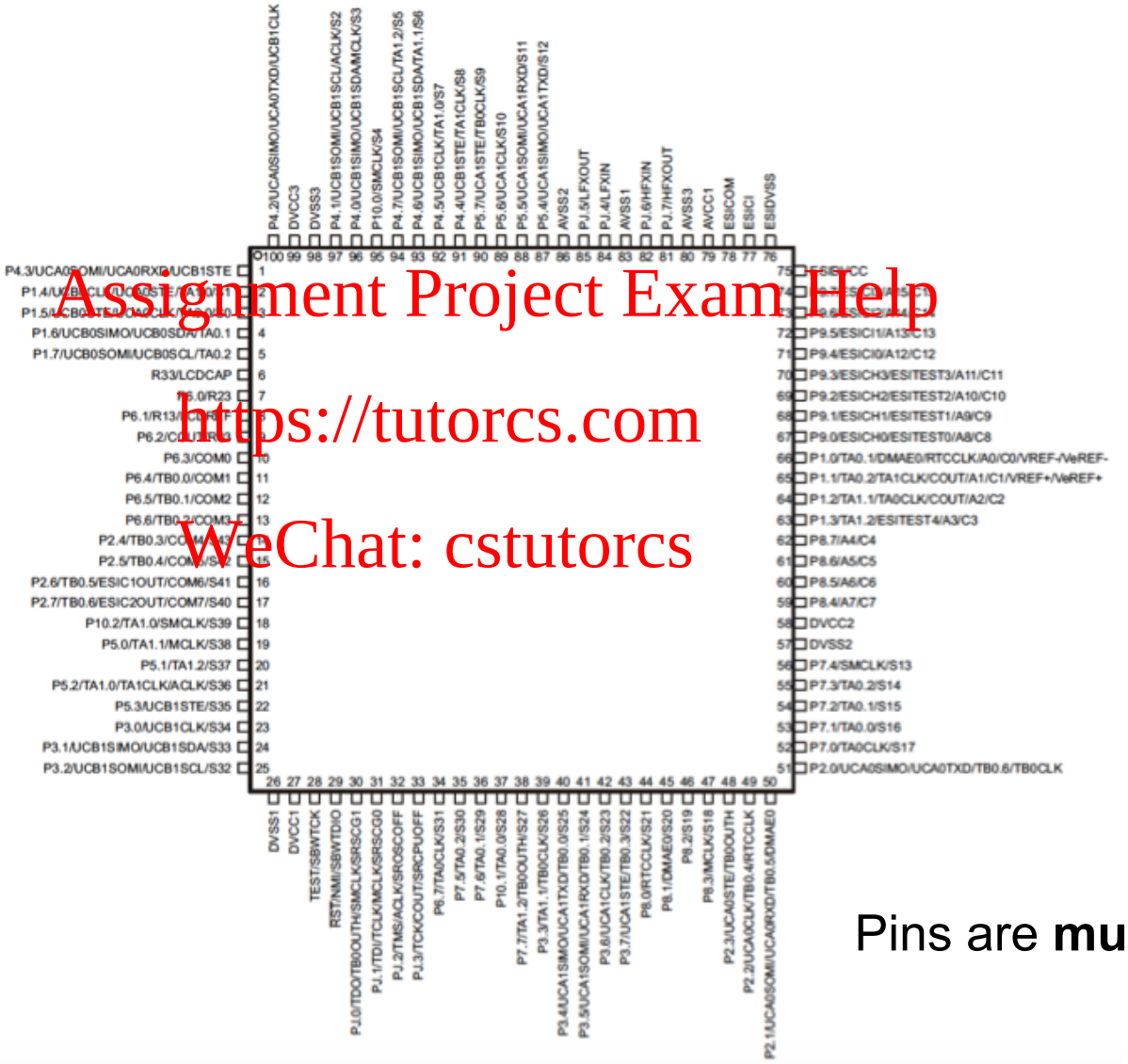
If each individual byte is addressed, we can address **65,536 B = 64 KiB**

The MSP430FR6989 Launchpad



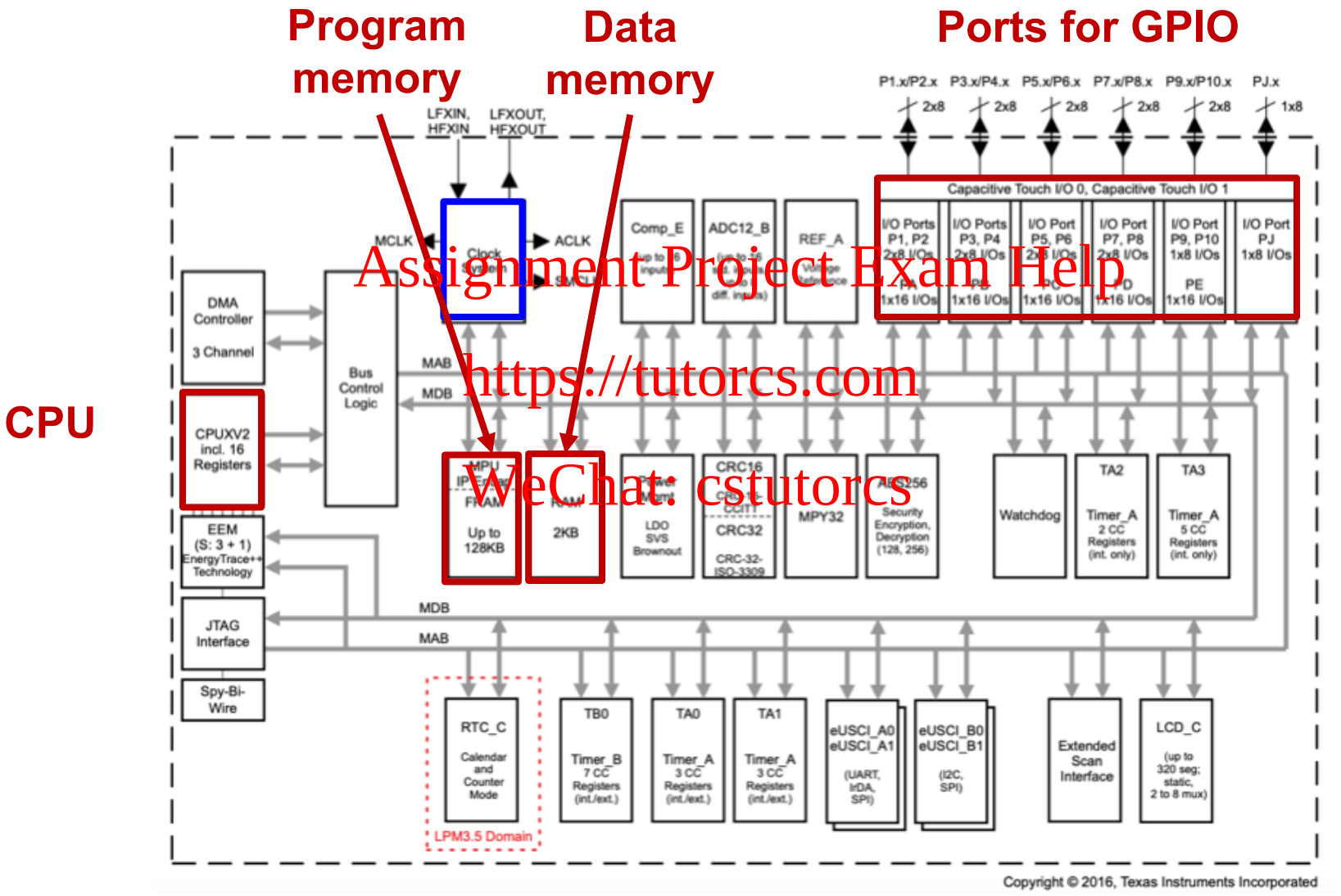
MSP430FR6989IPZ Pinout

slas789d.pdf



Pins are **multiplexed**

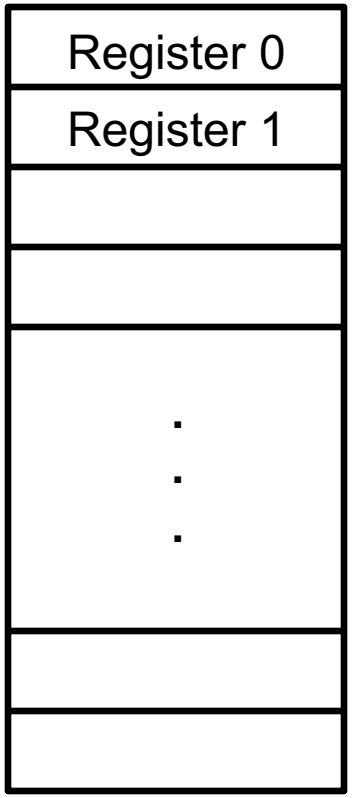
Inside MSP430FR6989



Memory Map

We can view the memory as a pile of **registers**

Each register holds 8 bits = 1 byte



These registers reside in different locations

- RAM registers (Data memory – 2048 B)
- FRAM registers (Program memory – 48 kB)
- 8-bit peripheral registers (GPIO P1-P10)
- 16-bit peripheral registers

Every register (except the 16 core registers in the CPU) is mapped to a **unique 16-bit memory address**

⇒ **Memory mapping**

This is a design choice – all registers are on the **same address bus**

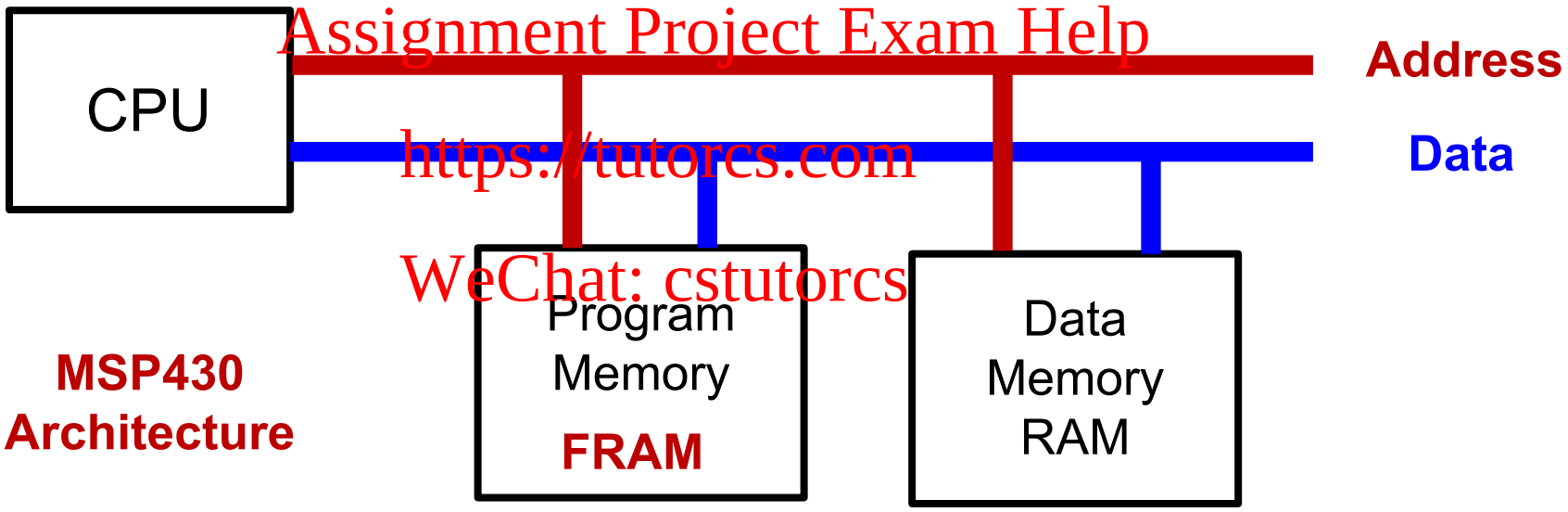
There are two different architectures

- von Neumann Architecture
- Harvard Architecture

von Neumann Architecture

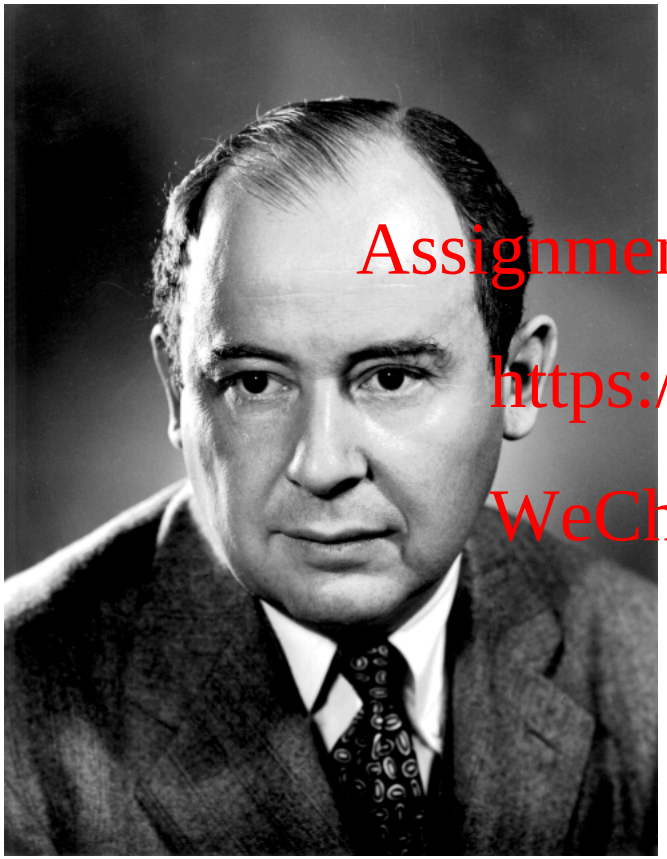
In the **von Neumann Architecture** (aka **Princeton Architecture**) the one data and one address bus serve both the program and the data memory

⇒ There is one set of addresses



Same data and address bus and **same** set of addresses for program and data memory and all other peripheral registers

John von Neumann



1903 – 1957

Hungarian-American polymath

- Mathematician
- Computer scientist
- Quantum physicist
- Economist (Game theory)
- Engineer

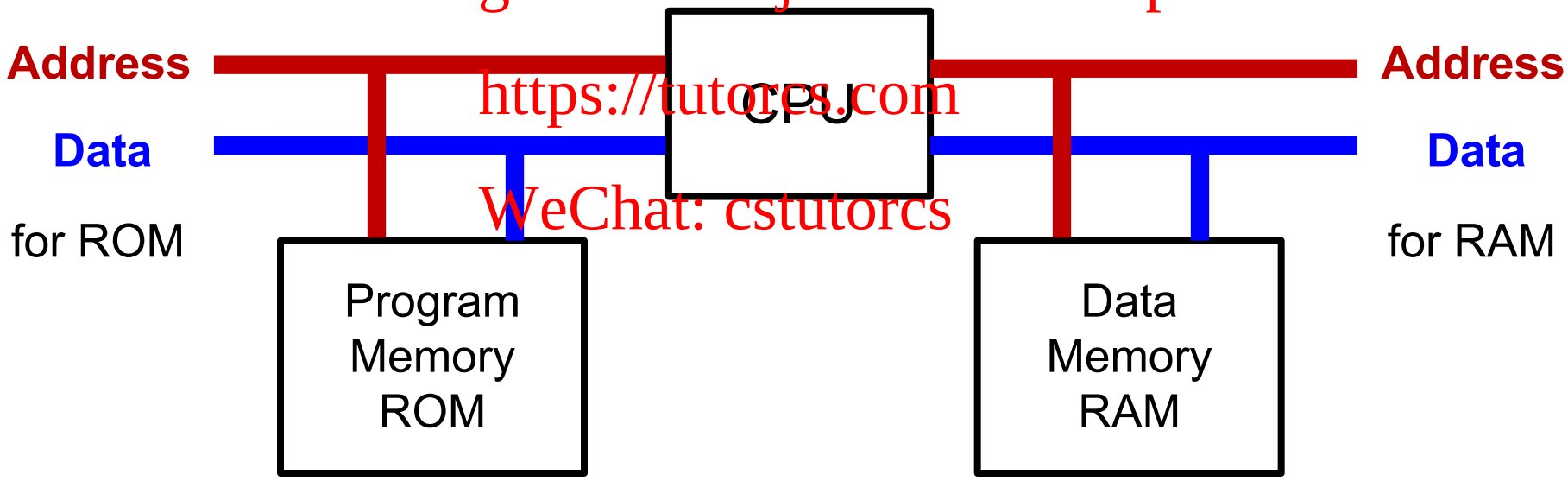
Institutions

- Princeton University
- Manhattan Project
- US Atomic Energy Commission

Harvard Architecture

In the **Harvard Architecture** the program and data memory are served by different data and address busses

⇒ Program and data memory can have different address sets, widths etc.



There are also separate control busses serving RAM and ROM

v. Neumann vs. Harvard Architecture

Harvard Architecture

Efficient

- Simultaneous access to program and data memory

But complex

- Constants (in program memory) and variables (in data memory) live in different address spaces and must be treated differently

von Neumann Architecture

Less efficient

- Program and data memory must be accessed one at a time
- von Neumann bottleneck

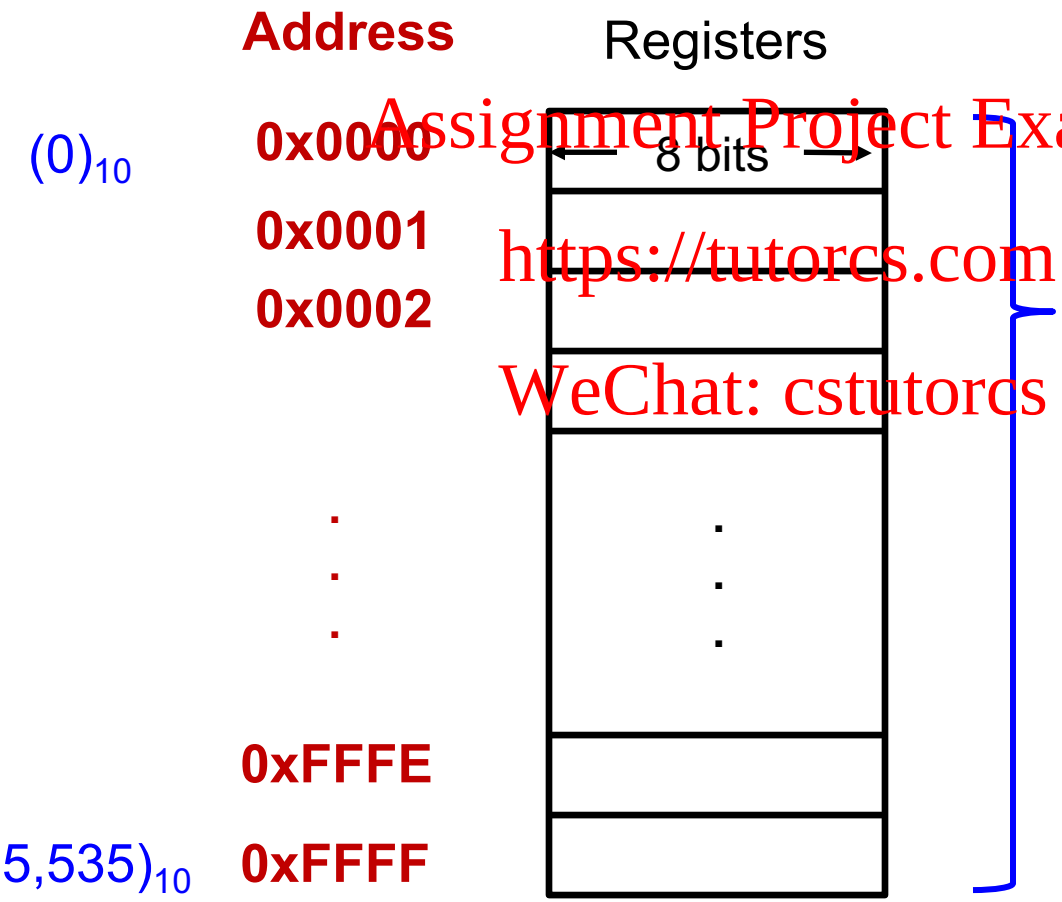
Simpler

- Constants (in program memory) and variables (in data memory) are addressed in the same way

Memory Map

The address bus is 16 bits wide, i.e., each address is 16 bits

Addresses are expressed as **Hex numbers from 0x0000 to 0xFFFF**



Each byte of memory is addressed individually

total number of registers that can be addressed

$= 2^{16} = 65,536$

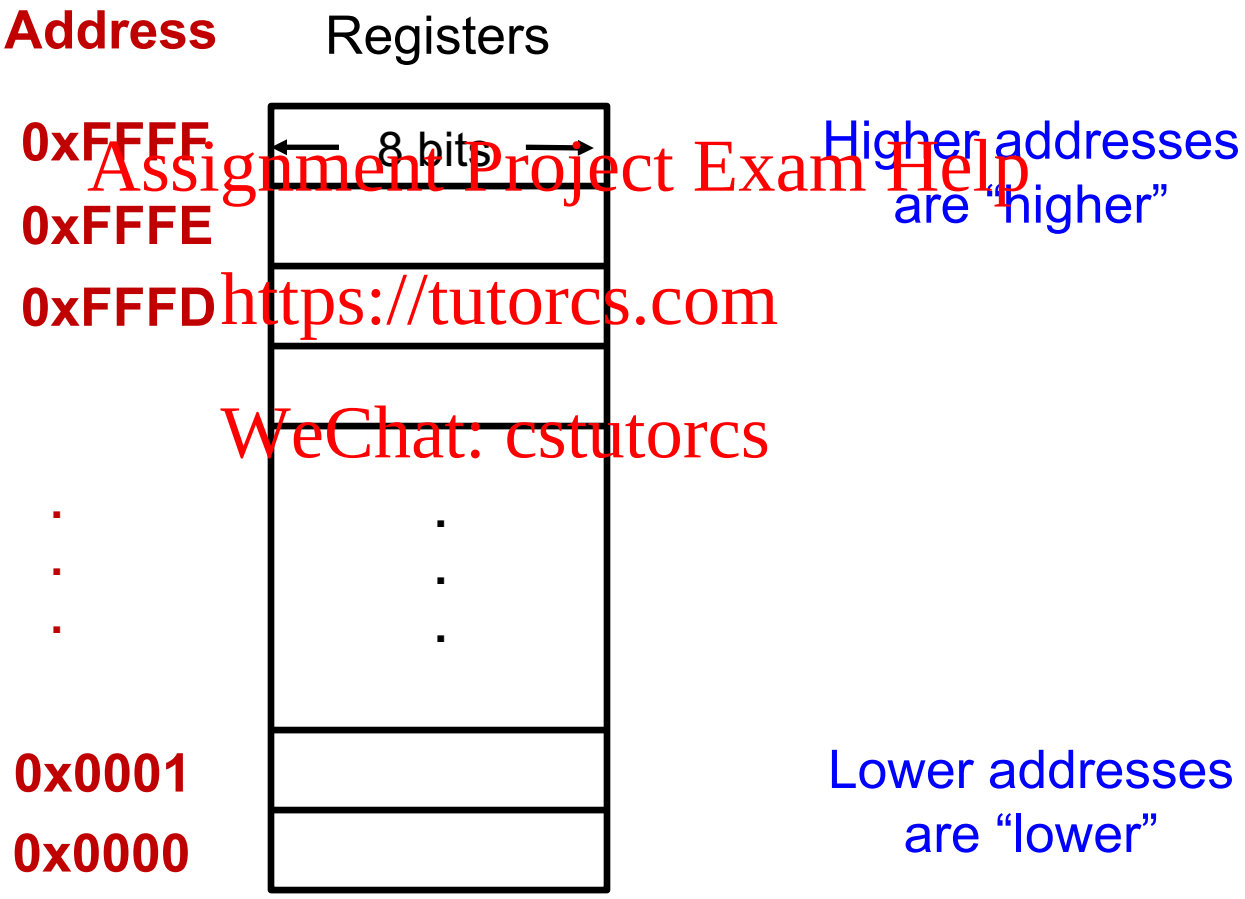
amount of memory that can be addressed

$= 65,536$ B

= 64 KiB

Memory Map

Some sources draw the map upside down e.g., TI uses this style



Word Addressing

The **data bus** is 16 bits wide ⇒ 16-bit architecture

The data bus can transfer either

- a **byte** (i.e., 8 bits)

or

- a **word** of 16 bits

Each byte is addressed

Each address corresponds to a byte

Not so fast! What is the address of a word???

The **address of a word** is address of the byte with the lower address

⇒ the address of a word is always **even**

- Bytes with addresses 0x0000 and 0x0001 => word with address **0x0000**
- Bytes with addresses 0x0002 and 0x0003 => word with address **0x0002**
- …
- Bytes with addresses 0xFFFE and 0xFFFF => word with address **0xFFFE**

Bytes w/ addresses 0x0001 and 0x0002 cannot be accessed as a word

Endiannes

Bytes with addresses 0x0000 and 0x0001 => word with address 0x0000

How is the word (two bytes) ordered over these two addresses?

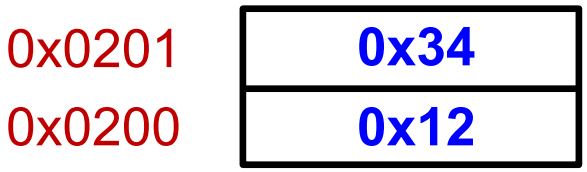
e.g.,

word **0x1234** **0x12** more/higher significant byte (MSB or HSB)

0x34 lower significant byte (LSB)

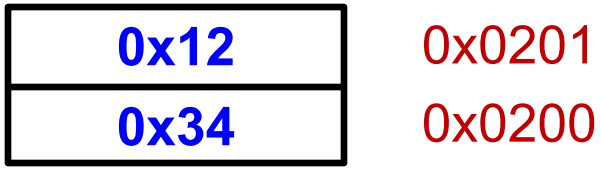
There are two ways of storing these bytes over two addresses

HSB at lower address
LSB at higher address



Big-Endian Ordering

HSB at higher address
LSB at lower address



Little-Endian Ordering

Endiannes

How is the word (two bytes) ordered over these two addresses?

e.g.,

word **0x1234** **0x12** more/higher significant byte (MSB or HSB)

0x34 lower significant byte (LSB)

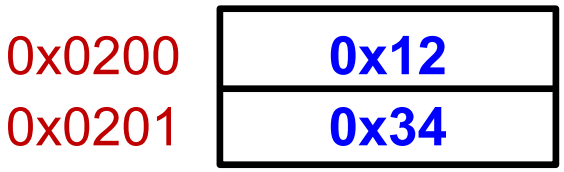
There are two ways of storing these bytes over two addresses

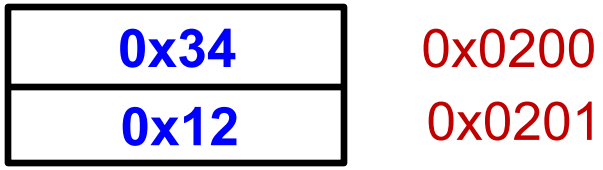
Big-Endian Ordering

HSB at lower address
LSB at higher address

Little-Endian Ordering

HSB at higher address
LSB at lower address



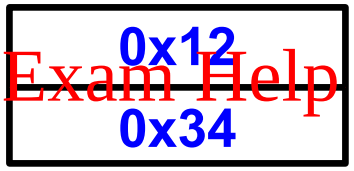


Little-Endian Ordering

The MSP430 uses **little-endian ordering** – more common format today

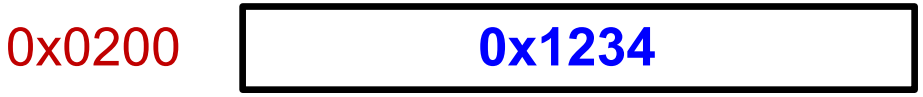
word **0x1234** is stored in memory as

HSB at higher address
LSB at lower address



In CCS addresses increase from left to right **when displaying bytes**

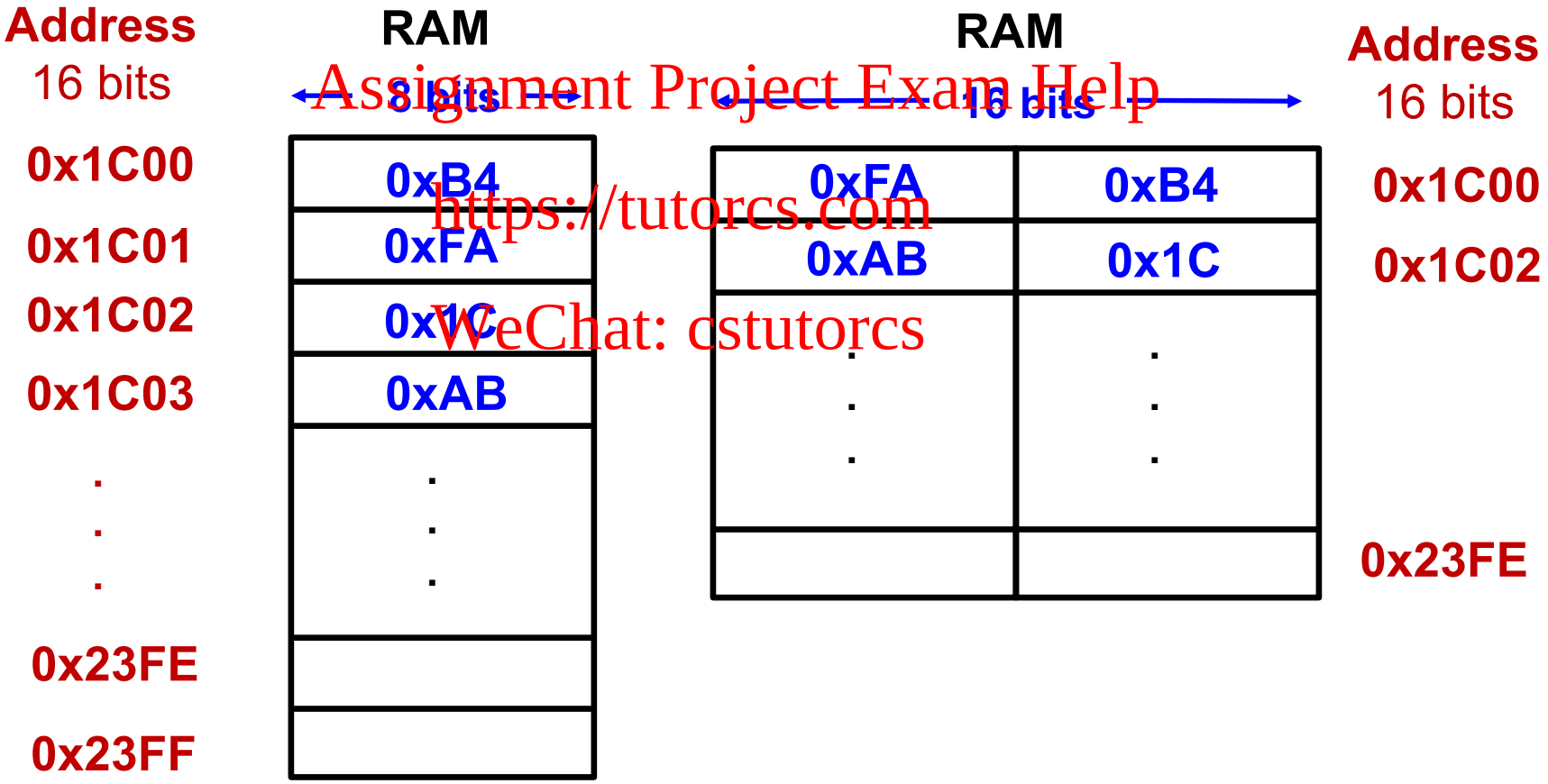
No issue when **displaying words**



Memory Mapping – RAM

The **RAM** is the data memory – stores run-time variables and the **stack**

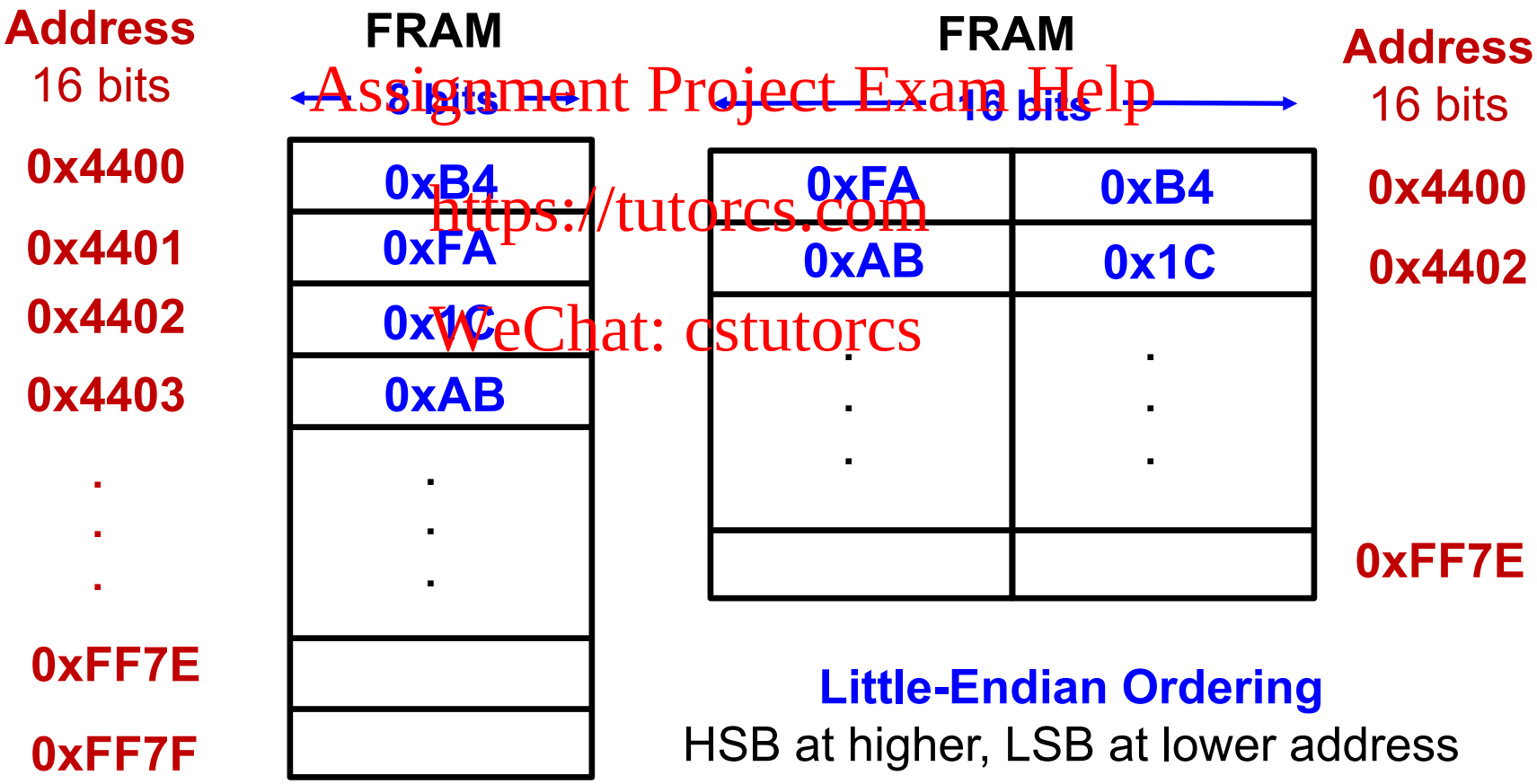
Size: 2048 B = 2 KiB each byte is addressed – memory mapped



Memory Mapping – FRAM

The **FRAM** (**F**erroelectric **RAM**) is the program memory of MSP430**FR**6989

Size: 48,000 B = 48 kB each byte is addressed – memory mapped



Little-Endian Ordering

HSB at higher, LSB at lower address

Memory Map

The linker file lnk_msp430fr6989.cmd contains the memory map

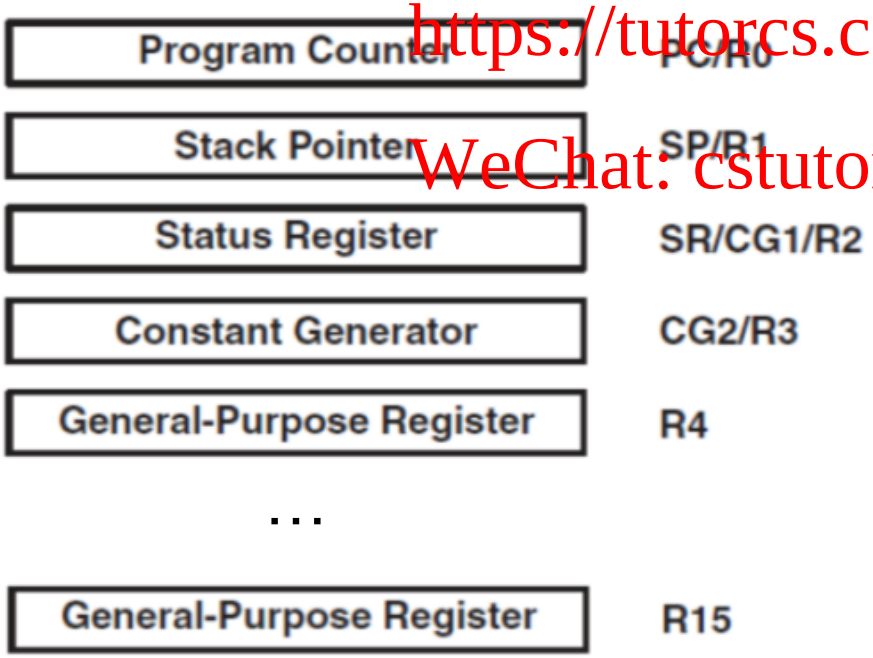
```
MEMORY
{
    TINYRAM                 : origin = 0x0006, length = 0x001A
    PERIPHERALS_8BIT        : origin = 0x0020, length = 0x00E0
    PERIPHERALS_16BIT       : origin = 0x0100, length = 0x0100
    RAM                     : origin = 0x1C00, length = 0x0800
    INFOA                   : origin = 0x1980, length = 0x0080
    INFOB                   : origin = 0x1900, length = 0x0080
    INFOC                   : origin = 0x1880, length = 0x0080
    INFOD                   : origin = 0x1800, length = 0x0080
    FRAM                    : origin = 0x4400, length = 0xBB80
    FRAM2                   : origin = 0x10000,length = 0x13FF8 /* Boundaries cha
    JTAGSIGNATURE           : origin = 0xFF80, length = 0x0004, fill = 0xFFFF
    BSLSIGNATURE            : origin = 0xFF84, length = 0x0004, fill = 0xFFFF
    IPESIGNATURE            : origin = 0xFF88, length = 0x0008, fill = 0xFFFF
    INT00                   : origin = 0xFF90, length = 0x0002
    INT01                   : origin = 0xFF92, length = 0x0002
    INT02                   : origin = 0xFF94, length = 0x0002
    INT03                   : origin = 0xFF96, length = 0x0002
    INT04                   : origin = 0xFF98, length = 0x0002
    INT05                   : origin = 0xFF9A, length = 0x0002
    INT06                   : origin = 0xFF9C, length = 0x0002
    INT07                   : origin = 0xFF9E, length = 0x0002
    INT08                   : origin = 0xFFA0, length = 0x0002
    INT09                   : origin = 0xFFA2, length = 0x0002
```

The 16 Core Registers

The 16 **core registers** R0 – R15 are 16-bit registers that

- sit in the CPU
- are not memory mapped – no memory address
- **access them directly by their name R4 – R15**

The first four core registers have dedicated functions



Announcements

Will post Quiz 2 by the end of tomorrow

- **You will have one week to solve all questions and submit**
- MCU architecture: CPU, Program/Data Memory, Address and Data Bus
- von Neumann and Harvard Architectures
- Byte and word addressing
- Little-Endian and Big-Endian ordering
- Memory map of MSP430FR6989 – RAM and FRAM