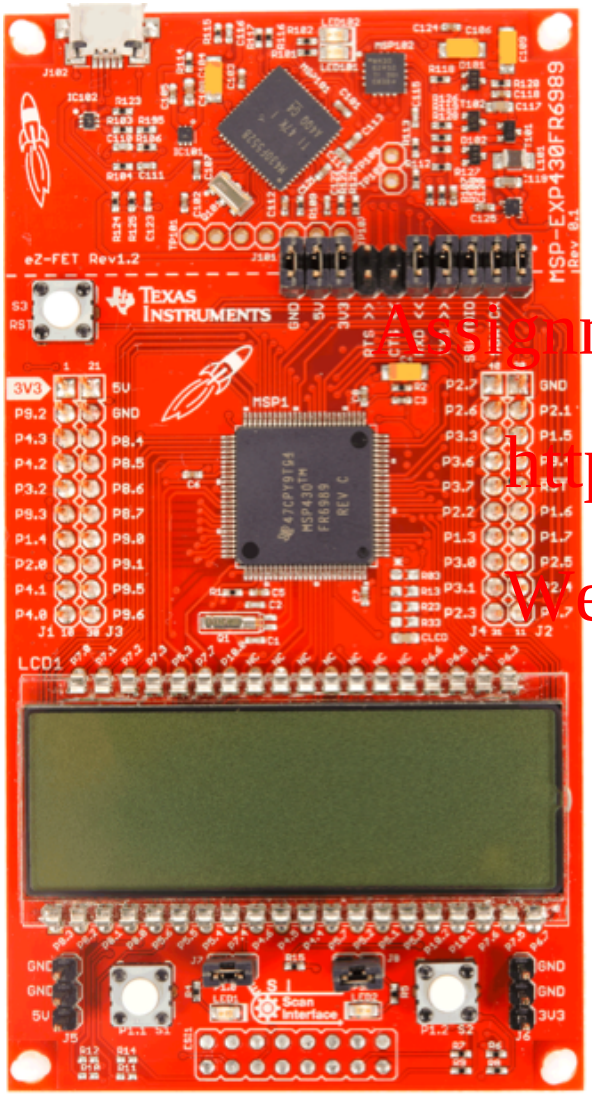


Lecture 4

Operations w/ Signed and Unsigned Numbers

***

Inside an MCU

Quiz #1

Posted to Carmen due Wednesday 11/25 before class – 4:10 pm

What do you need to know?

- Be fluent with binary, decimal and hexadecimal numbers and conversions
- Be fluent with signed numbers using two's complement notation
- Know how to take 2's complement
- Understand how addition/subtraction works for signed/unsigned numbers
- Understand when overflow occurs with signed/unsigned numbers
- Know how to multiply a signed/unsigned number by a power of 2
- Know how to divide a signed/unsigned number by a power of 2
- Be aware of the shortcomings with multiplication/division as ⬆️

Today

- Make sure you can do all this with 8-bit and 16-bit arithmetic

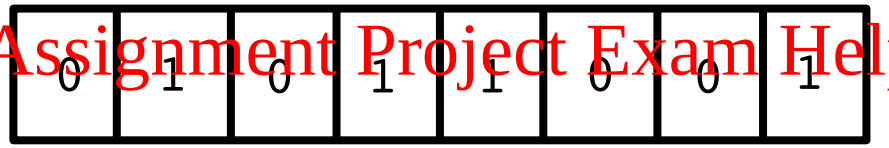
Last Time: Signed Numbers w/ 2's Complement

Modern computers use **2's complement representation** for signed numbers

Positive numbers: use binary representation of the number

e.g.,

89

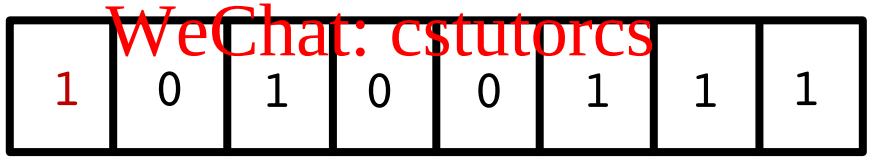


in Hex 0x59

Negative numbers: use two's complement of the absolute value

e.g.,

- 89



in Hex 0xA7

8-bits can represent 256 distinct values

- 128 of them will be positive or zero – do not complement 0 to 127
- 128 of them will be negative – use two's complement -128 to -1

Need to partition so that there is **no overlap** between both sets

Last Time: Signed Numbers w/ 2's Complement

8-bit signed numbers range from -128 to 127

n-bit signed numbers
– 2^{n-1} to $2^{n-1} – 1$

not sign-magnitude

Binary	Decimal	
1000 0000	-128	**negative numbers** start with a "1"
1000 0001	-127	
...		
1111 1110	-2	
1111 1111	-1	
0000 0000	0	**positive numbers** start with a "0"
0000 0001	1	
...		
0111 1110	126	
0111 1111	127	

2's complement of the absolute value

binary representation of the number

For hexadecimal signed numbers convert binary to hex: -2 is 0xFE

Problem of Overflow I

Part 1: Overflow in case of unsigned numbers

8-bit register can hold unsigned numbers from 0 to 255

If the sum of two unsigned numbers is greater than 255 we have **overflow**

– the resulting sum does not fit into the 8-bit register


```
  01101011        107
+ 11010010      + 210
----------      -----
100111101        317
```

overflow!

value inside register: 61 **wrong!**

Problem of Overflow II

Part 2: Overflow in case of signed numbers

8-bit register can hold signed numbers from –128 to 127

We have **overflow** when

- the sum of two positive numbers is greater than 127
- the sum of two negative numbers is less than –128

107	01101011	107
+ 82	+ 01010010	+ 82
189	10111101	–67
Unsigned Number Interpretation **No overflow**		**wrong!** **overflow!**

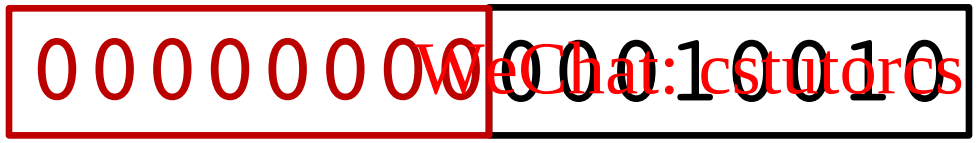
Padding Signed Numbers

Overflow $\Rightarrow$ Need to work with larger sized registers

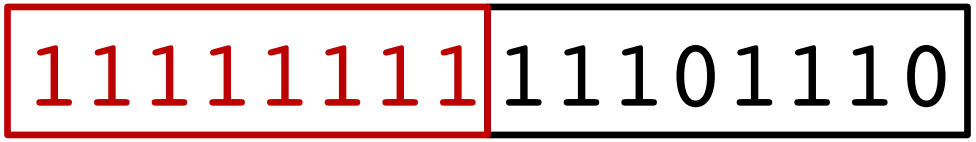
How do we go from 8-bit signed numbers to 16-bit signed numbers?

Depends on the sign of the number

If the number is **positive** pad with **zeros**



If the number is **negative** pad with **ones**



$= (65518)_{10}$ Two's complement: $2^{16} - 65518 = 18$

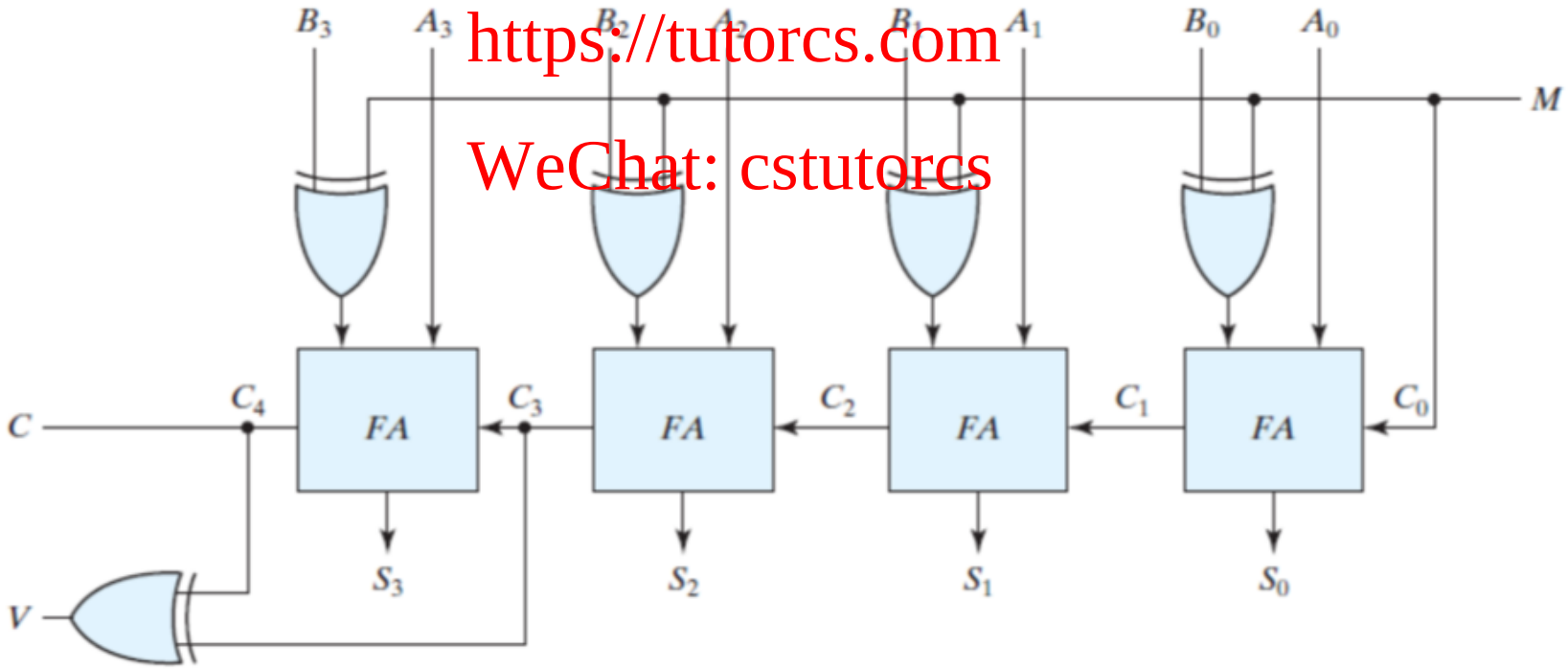
Subtraction

Subtraction is easy once we have figured out negative numbers and addition

Why?

A – B = A + (2's complement of B)

4-bit adder/subtractor: adds when M = 0, subtracts when M = 1



Multiplication and Division

Our MCU has instructions for addition and subtraction

– but not for multiplication or division

Multiplication/division is much more expensive than addition/subtraction

- more gates
- more cycles
- more power consumption

We **cannot** easily multiply by arbitrary numbers

– **but multiplication by a power of two is very easy!**

How do we multiply by 10 in base-10?

Easy! Append a 0.

Shift all digits to the left,

How do we multiply by 2 in base-2?

Shift all bits to the left, append a 0

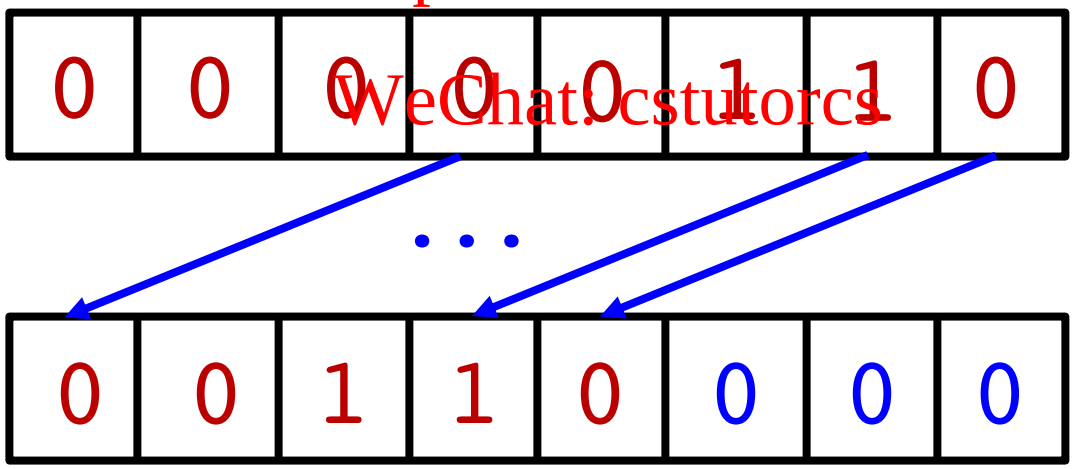
3	x	2	=	6
II				II
$(11)_2$				$(110)_2$

Multiplication by a Power of Two

To multiply a binary number N by 2^m

- **Shift the number m bits to the left and append with zeros**
- Make sure that there is no overflow!!

e.g.: 6 x 8 $8 = 2^3 \Rightarrow$ Shift 3 bits to the left



32 + 16 = 48

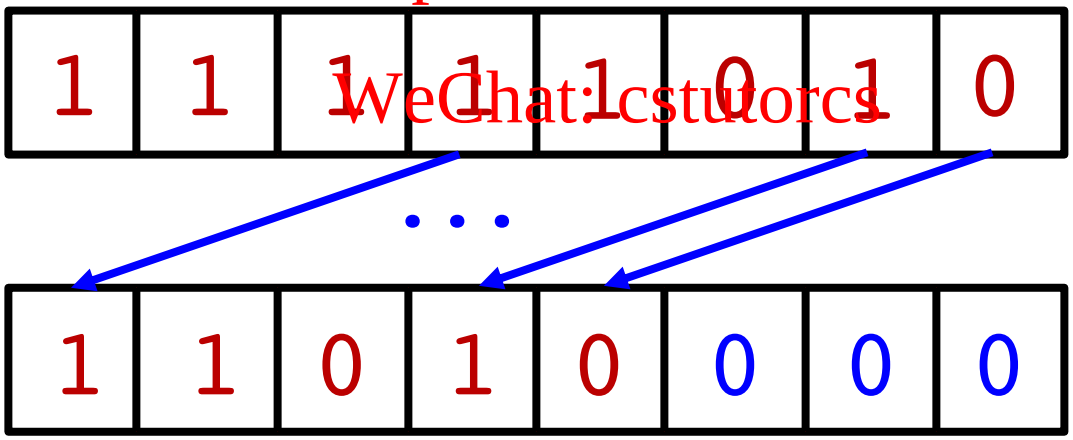
Multiplication by a Power of Two

To multiply a binary number N by 2^m

- **Shift m-bits to the left and append with zeros**
- Make sure there is no overflow!!

Does this work with signed numbers too?

e.g.: – 6 x 8 $\quad 8 = 2^3 \quad \Rightarrow$ Shift 3 bits to the left



2's comp't $\quad 0\ 0\ 1\ 1\ 0\ 0\ 0\ 0$

Magic!

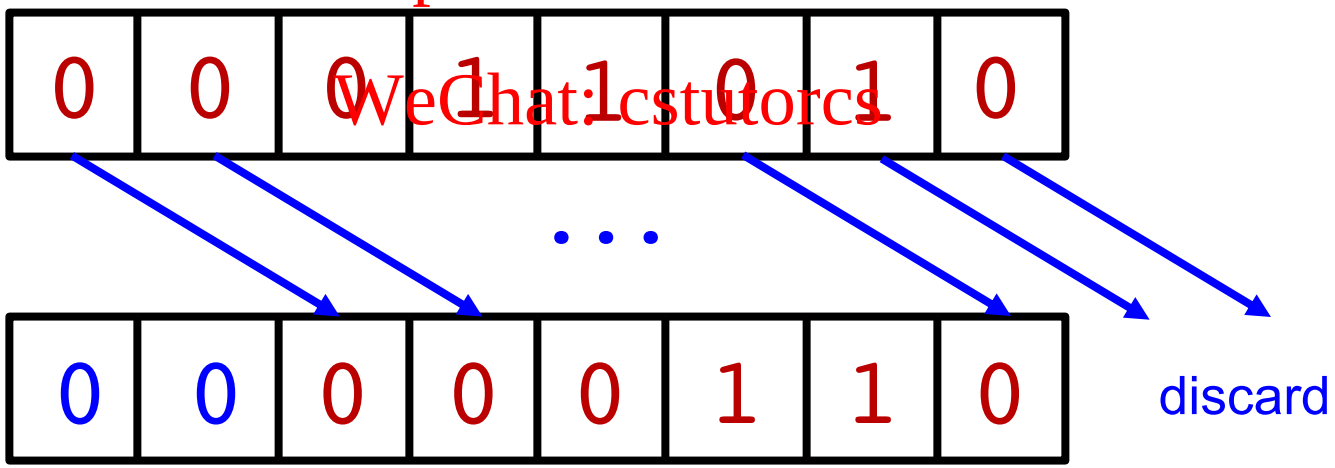
$\Rightarrow$ **- 48**

Division by a Power of Two

To divide an **unsigned number** by 2^m

- **Shift m-bits to the right and pad with zeros**
- The answer will not be exact – we are discarding the fractional part

e.g.: 26 ÷ 4 $\quad 4 = 2^2 \Rightarrow$ Shift 2 bits to the right



$\Rightarrow$ **6**

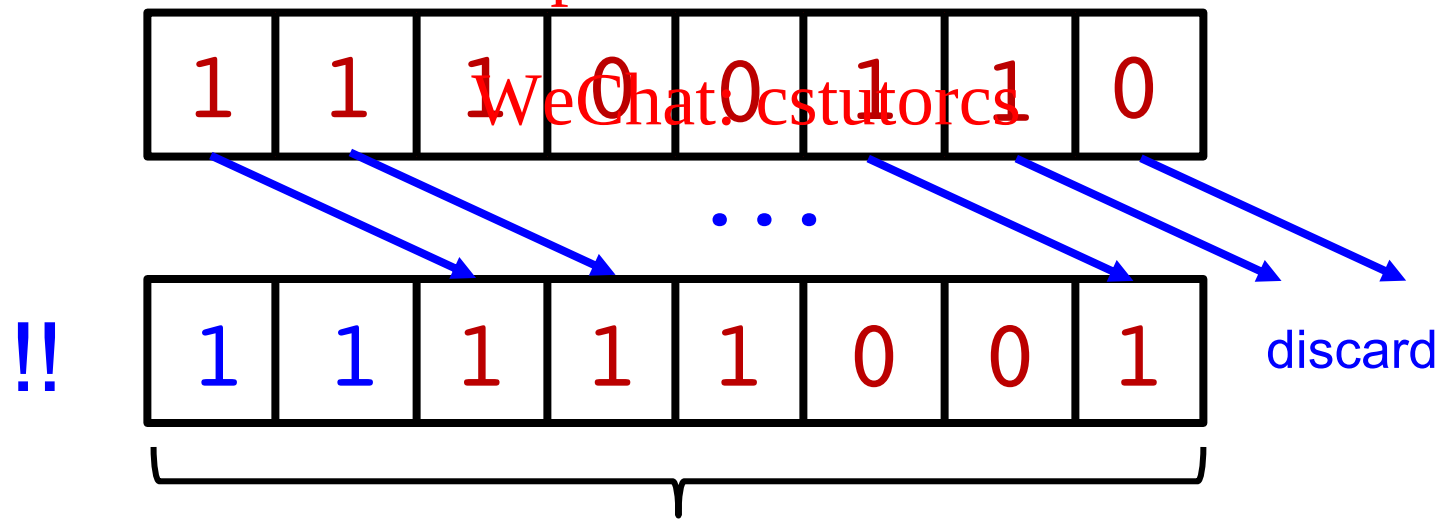
Division by a Power of Two

To divide a **signed number** by 2^m

- **Shift m-bits to the right & pad with the most significant bit "sign bit"**
- The answer will not be exact – we are discarding the fractional part

taking the floor function: $\lfloor -6.5 \rfloor = -7$

e.g.: –26 ÷ 4 $4 = 2^2 \Rightarrow$ Shift 2 bits to the right



249 in decimal $\Rightarrow$ **- 7**

Microcontrollers (MCU)

At the beginning we have the microprocessor

A **microprocessor** contains a complete digital processor including at least an **arithmetic logic unit (ALU)** and associated **registers**

e.g., Intel 4004

4-bit processor released in 1971 First commercial microprocessor

A microprocessor needs many other components to support it: external memory, I/O devices etc. e.g., the processor of a personal computer

A **microcontroller (MCU)** contains all of the functions to make a complete computer system on the same chip as the processor – including

- memory
- peripherals for I/O
- clock
- analog-to-digital converters

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

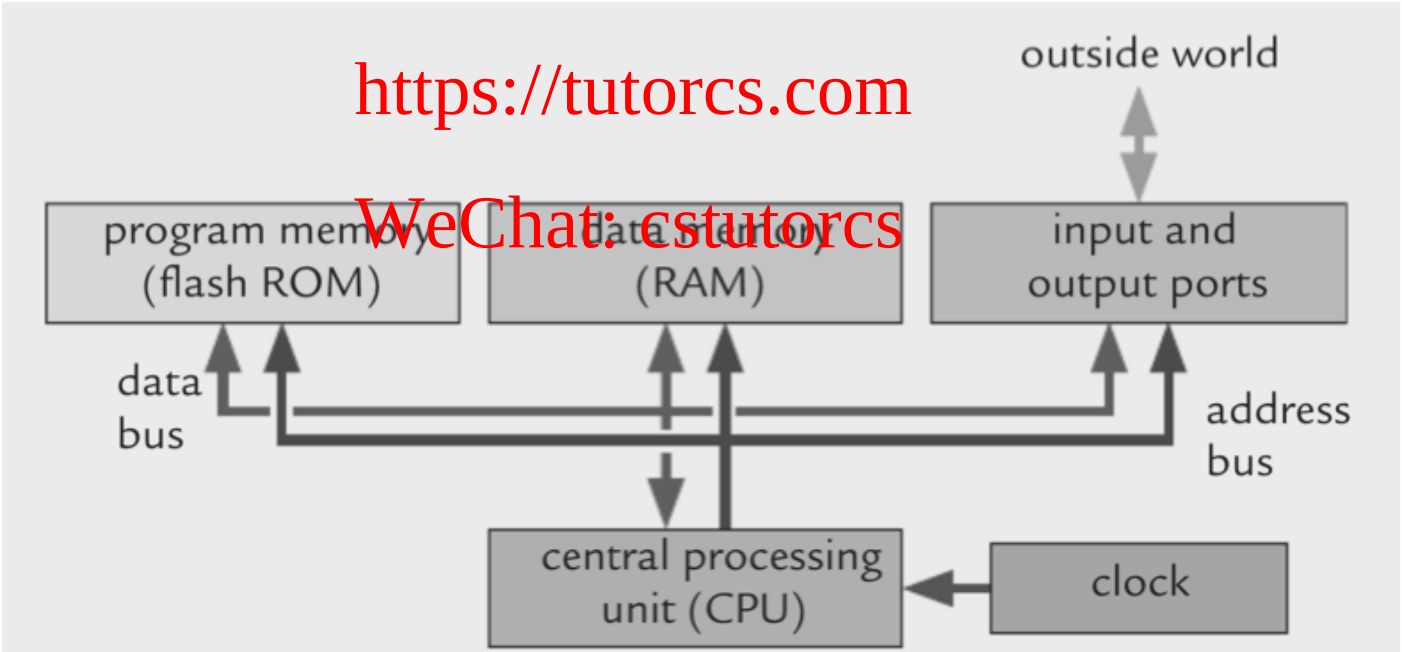


Figure 1.2: Essential components of a microcontroller.

Central Processing Unit

The **central processing unit (CPU)** includes

- Arithmetic Logic Unit (ALU) which performs the computations and logic

Instruction Set

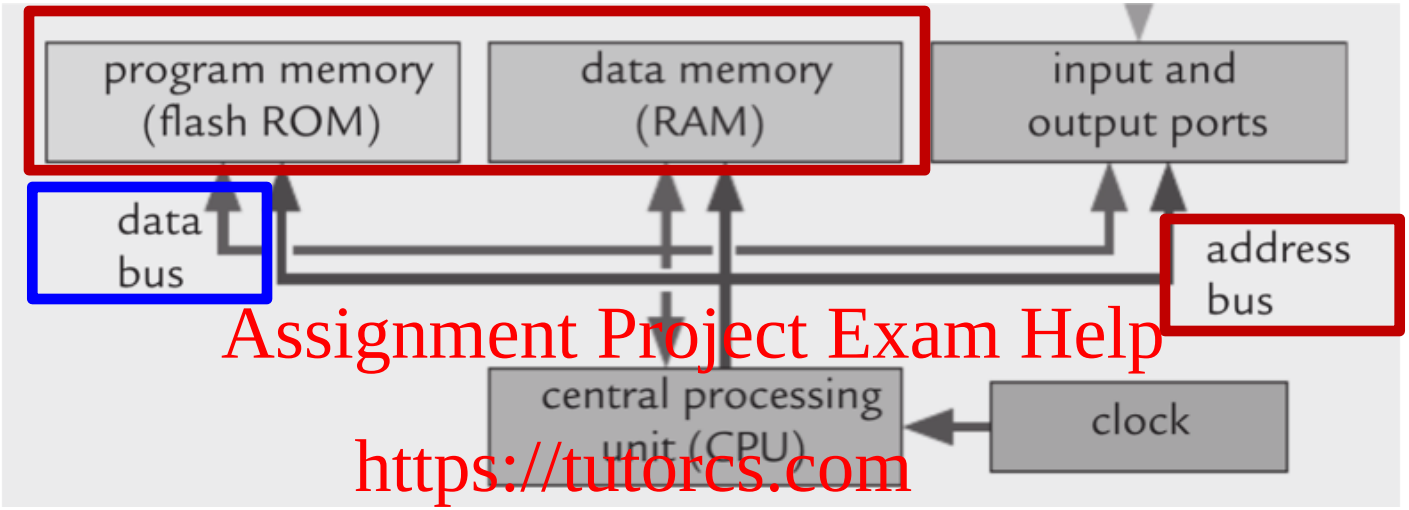
- Registers

Core Registers

  - Registers needed for the basic operation of the CPU
    - **Program Counter (PC)**
    - **Stack Pointer (SP)**
    - **Status Register (SR)**
  - Registers to hold operands and results

- Instruction decoder and other logic to control the CPU, handle resets and interrupts etc.

Memory



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., data and address memory and peripheral registers

Program and Data Memory

Program memory is where the machine code is stored

- Program memory needs to be **non-volatile**
  i.e., the stored information is retained after the power source is removed
  e.g., solid state drives (SSD), flash memory, Ferroelectric RAM (FRAM)
- In real world applications this can be **read-only memory (ROM)**
- But when developing code it needs to be erasable/rewritable
- Traditionally called **ROM** Our MCU 128000 bytes of usable FRAM **128 kB**

Data memory is where running code stores its data including the **stack**

- Data memory is usually volatile
  i.e., stored information is lost once the power source is removed
- Traditionally called **RAM – random access memory**
- RAM is very expensive (both silicon and power consumption)
- MCU have very small RAM Our MCU has only 2048 bytes of RAM **2 KiB**

Units of Memory

Memory is measured in **bits** and **bytes** **1 B = 1 byte = 8 bits**

What is a kilobyte?

Is it 1kB = 1000 B?
Is it 1kB = 1024 B?
Is it 1KB = 1024 B?

Is it 1MB = 1,000,000 B?
Is it 1MB = 1,048,576 B?

The SI unit prefix k (kilo) is always 1000!!! **1kB = 1000 B**

But, there is a reason for measuring in multiples of 1024 B
Hence the new prefix **"kilo binary"** or **kibi** written as **Ki**

1 KiB = 1024 B

1MiB = 1,048,576 B