

# Computer Architecture & Real-Time Operating System

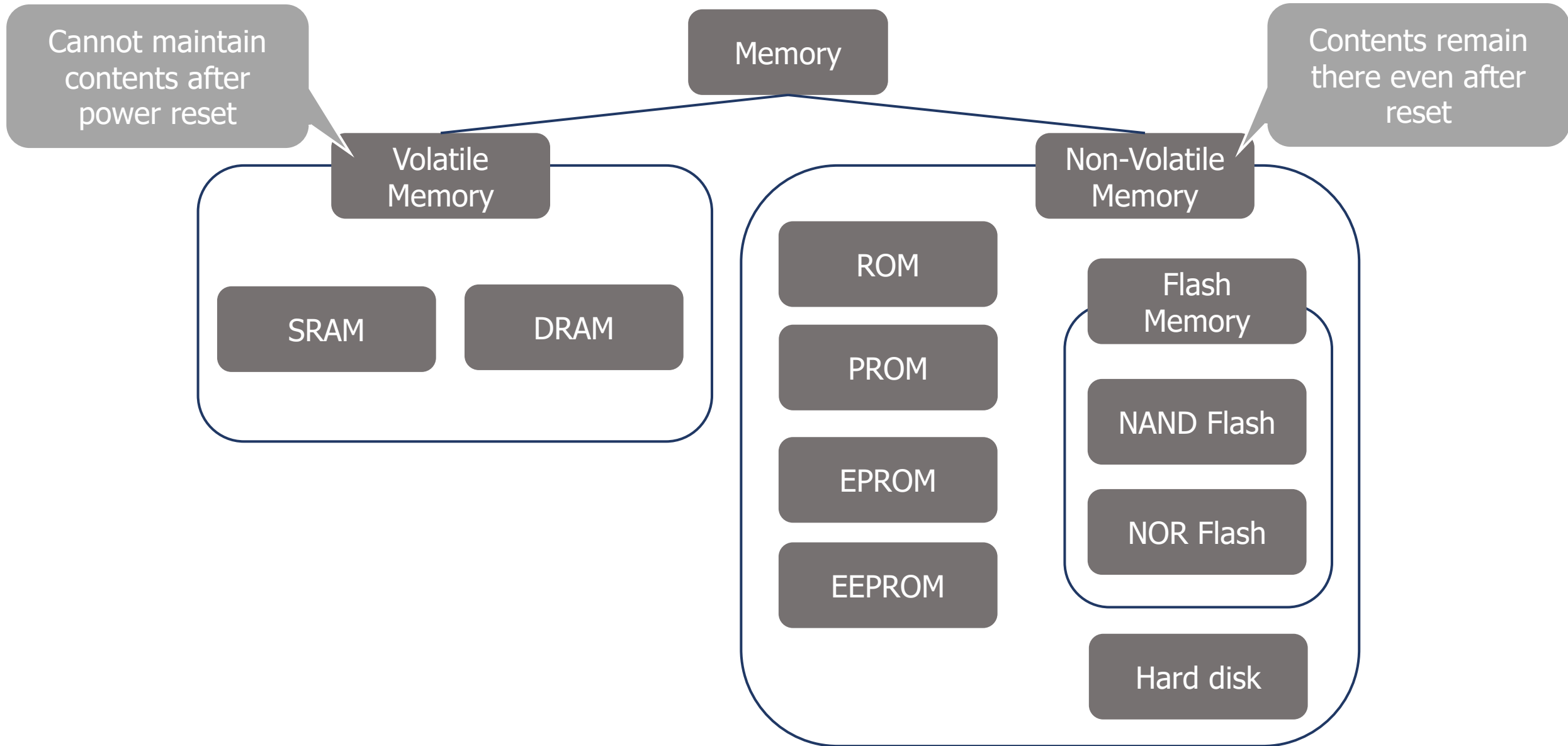
## 8. Memory Subsystem (1/2)

**Prof. Jong-Chan Kim**

**Dept. Automobile and IT Convergence**

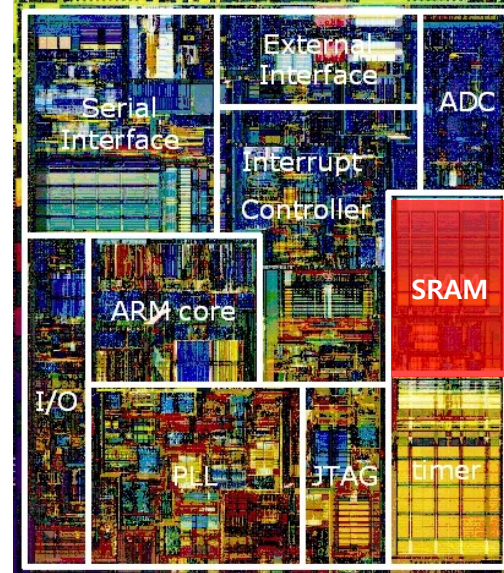
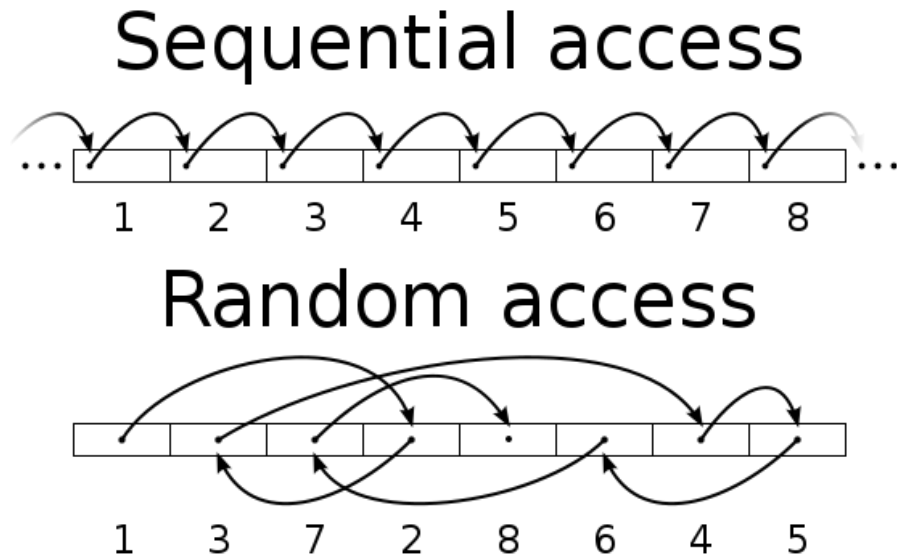


# Memory Technologies

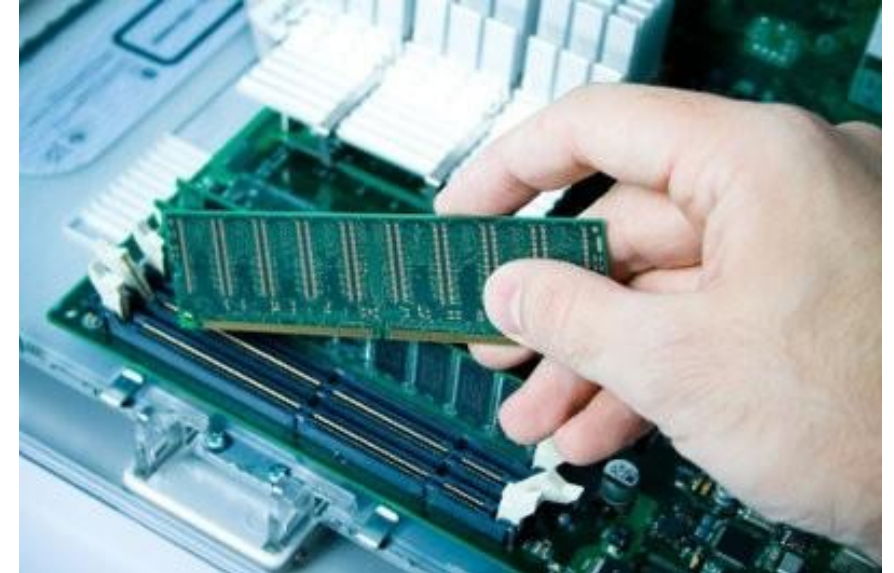


# RAM (Random Access Memory)

- Cannot maintain content after power reset
- Array of bytes, which is byte-addressable through the system bus



On-Chip SRAM

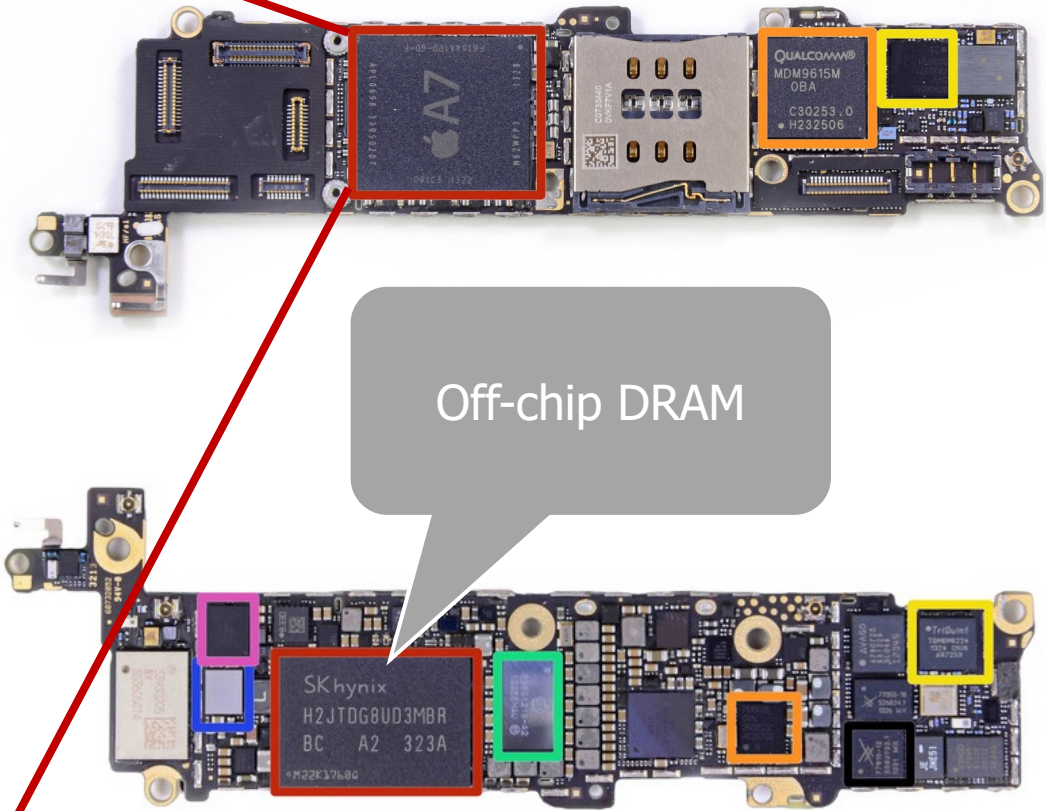
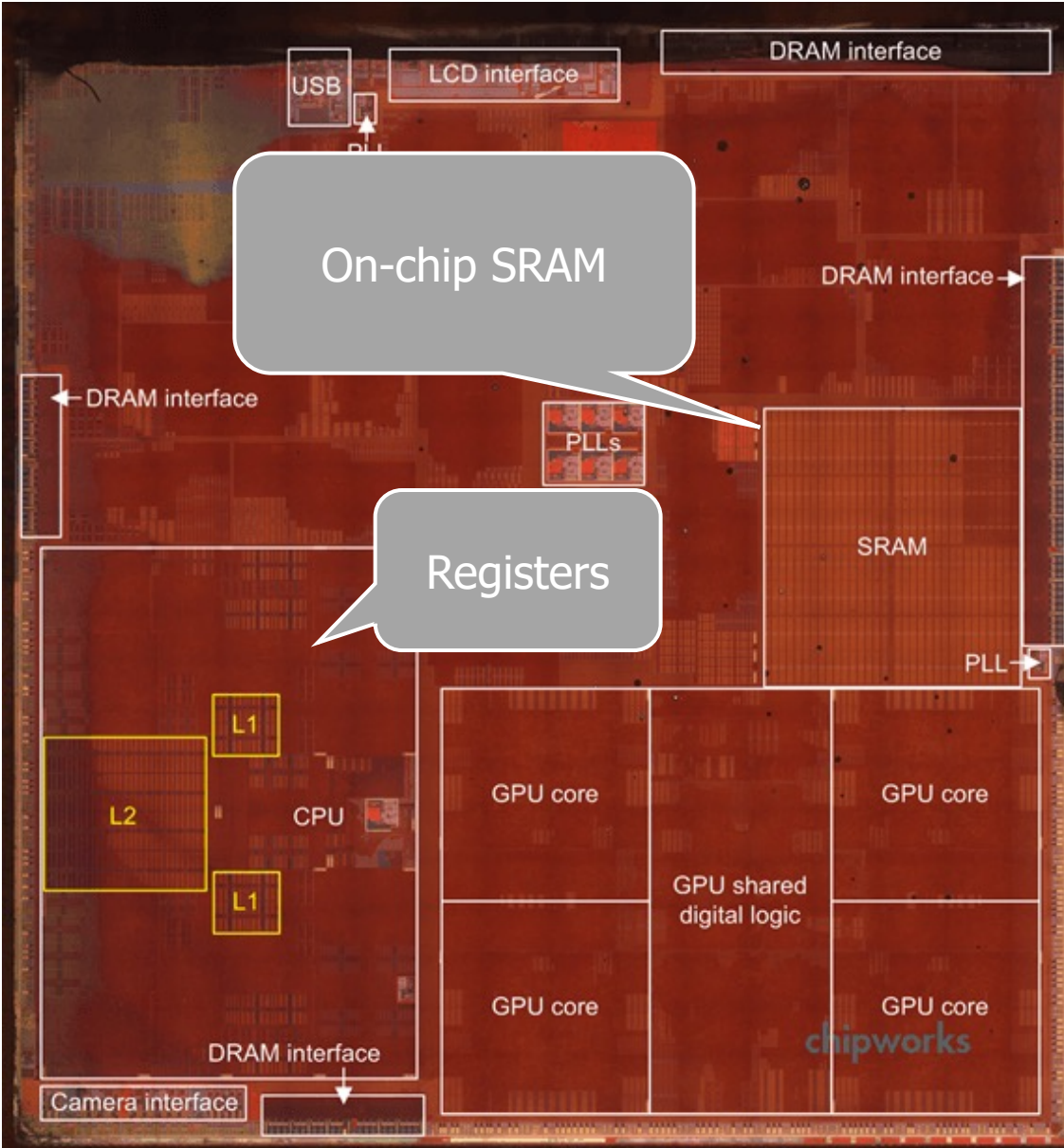


Off-Chip DRAM

# Two Types of RAM

SRAM (Static RAM)	DRAM (Dynamic RAM)
<ul style="list-style-type: none"><li>• Faster than DRAM (10x)</li></ul>	<ul style="list-style-type: none"><li>• Slower than SRAM (1x)</li></ul>
<ul style="list-style-type: none"><li>• Expensive (100x)</li></ul>	<ul style="list-style-type: none"><li>• Less Expensive (1x)</li></ul>
<ul style="list-style-type: none"><li>• No refresh required</li></ul>	<ul style="list-style-type: none"><li>• Refresh required</li></ul>
<ul style="list-style-type: none"><li>• Use less power</li></ul>	<ul style="list-style-type: none"><li>• Use more power</li></ul>
<ul style="list-style-type: none"><li>• Mostly used for on-chip cache or scratchpad memory</li></ul>	<ul style="list-style-type: none"><li>• Mostly used for off-chip main memory</li></ul>

# On-chip SRAM and Off-chip DRAM

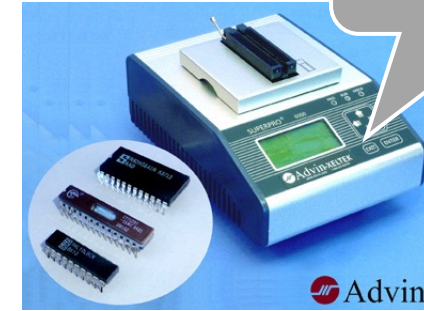




# Non-Volatile Memory

- ROM (Read Only Memory)
  - The content is hardwired while manufacturing
- PROM (Programmable ROM)
  - Programmable once in its lifetime
- EPROM (Erasable PROM)
  - Erasable (entirely) by exposing to UV light
- EEPROM (Electrically Erasable PROM)
  - Allows byte-level read and write
  - Writing a byte takes too much time (milliseconds)
  - Mostly used for storing configuration data

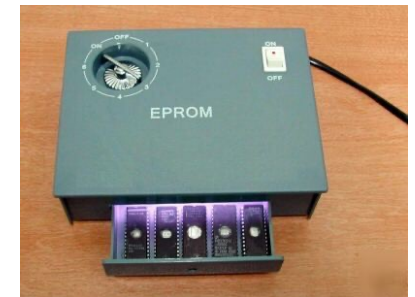
PROM  
Programmer



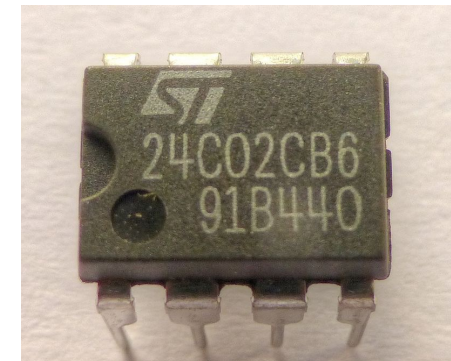
<http://www.advin.com/PROM-programmer.htm>



<http://electronics.stackexchange.com/questions/34607/erasing-eproms-with-sunlight>



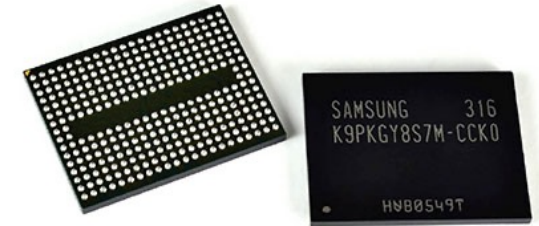
<http://electronics.stackexchange.com/questions/34607/erasing-eproms-with-sunlight>



<https://en.wikipedia.org/wiki/EEPROM>

# Non-Volatile Memory

- Flash Memory
  - Writable in pages ( $\sim$  KB)
  - Erasable in blocks ( $\sim$  MB)
  - Block must be erased before writing a page
  - Wear-leveling problem



NOR Flash	NAND Flash (used in SSDs and USB flash drives)
<ul style="list-style-type: none"><li>• Faster read</li></ul>	<ul style="list-style-type: none"><li>• Faster write</li></ul>
<ul style="list-style-type: none"><li>• Higher cost-per-byte</li></ul>	<ul style="list-style-type: none"><li>• Lower cost-per-byte</li></ul>
<ul style="list-style-type: none"><li>• Used for store instructions (text section)</li></ul>	<ul style="list-style-type: none"><li>• Used for store multimedia data</li></ul>
<ul style="list-style-type: none"><li>• Byte-level random access (memory mapped)</li></ul>	<ul style="list-style-type: none"><li>• No byte-level random access (port-based I/O)</li></ul>
<ul style="list-style-type: none"><li>• Use less power</li></ul>	<ul style="list-style-type: none"><li>• Use more power</li></ul>
<ul style="list-style-type: none"><li>• XIP (eXecute In Place)</li></ul>	<ul style="list-style-type: none"><li>• No XIP</li></ul>

# Example Memory Configurations

## Technical specs

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
LED_BUILTIN	
Length	68.6 mm
Width	53.4 mm
Weight	25 g

Data, Bss, Stack, ...

← Code

Long-term configuration data



<https://www.arduino.cc/en/Main/ArduinoBoardUno>

CPU Clock: 180 MHz

Including:

- TriBoard Hardware
- USB cable
- Power Supply
- Extension Board
- Getting Started, first 3 Steps to install the Tools, set up your Hardware, write and debug the first program
- Technical Documentation: e.g. User manuals (System unit and Peripheral unit), Architecture manual, Application notes, Data Sheets, Board Documentation (pdf-version)
- Evaluation Versions of Development Tools: e.g. Compiler, Debugger from Tool Partners

On-Chip Memory:

- 4 MByte embedded program flash with ECC,
- 16KByte EEPROM (emulated by 64KByte data Flash),
- 156 KByte on-chip SRAM,
- 4KByte data cache,
- 16 KByte instruction cache.

On-Boards Memory:

- Burst FLASH up to 16 MBytes (default: 4 MBytes),
- asynchronous SRAM up to 1MBytes (default),
- optional synchronous SRAM up to 8 MBytes.

Cache memory will be discussed in the next lecture

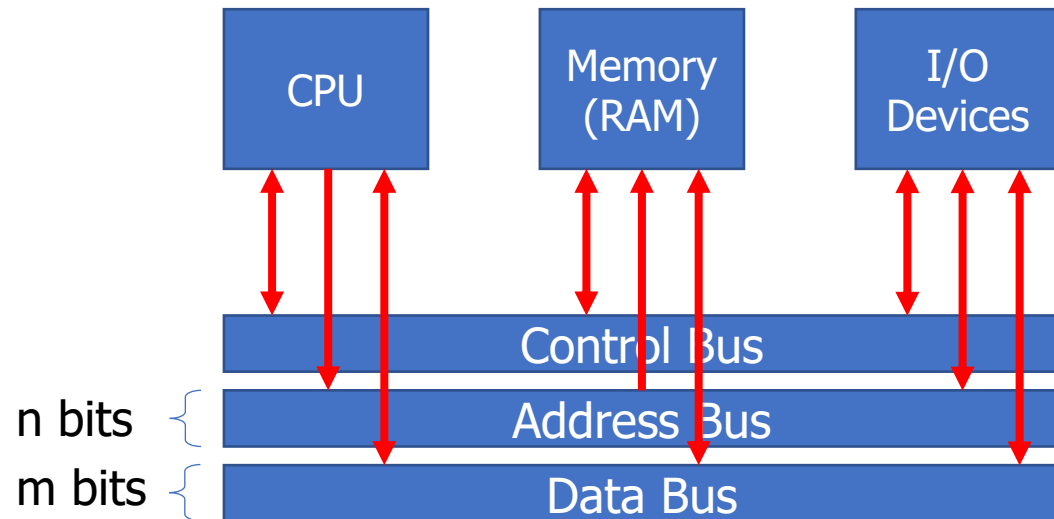


<http://www.ehitex.de/en/starter-kits/for-tricore/2111/starterkit-fr-tricore-tc1797>

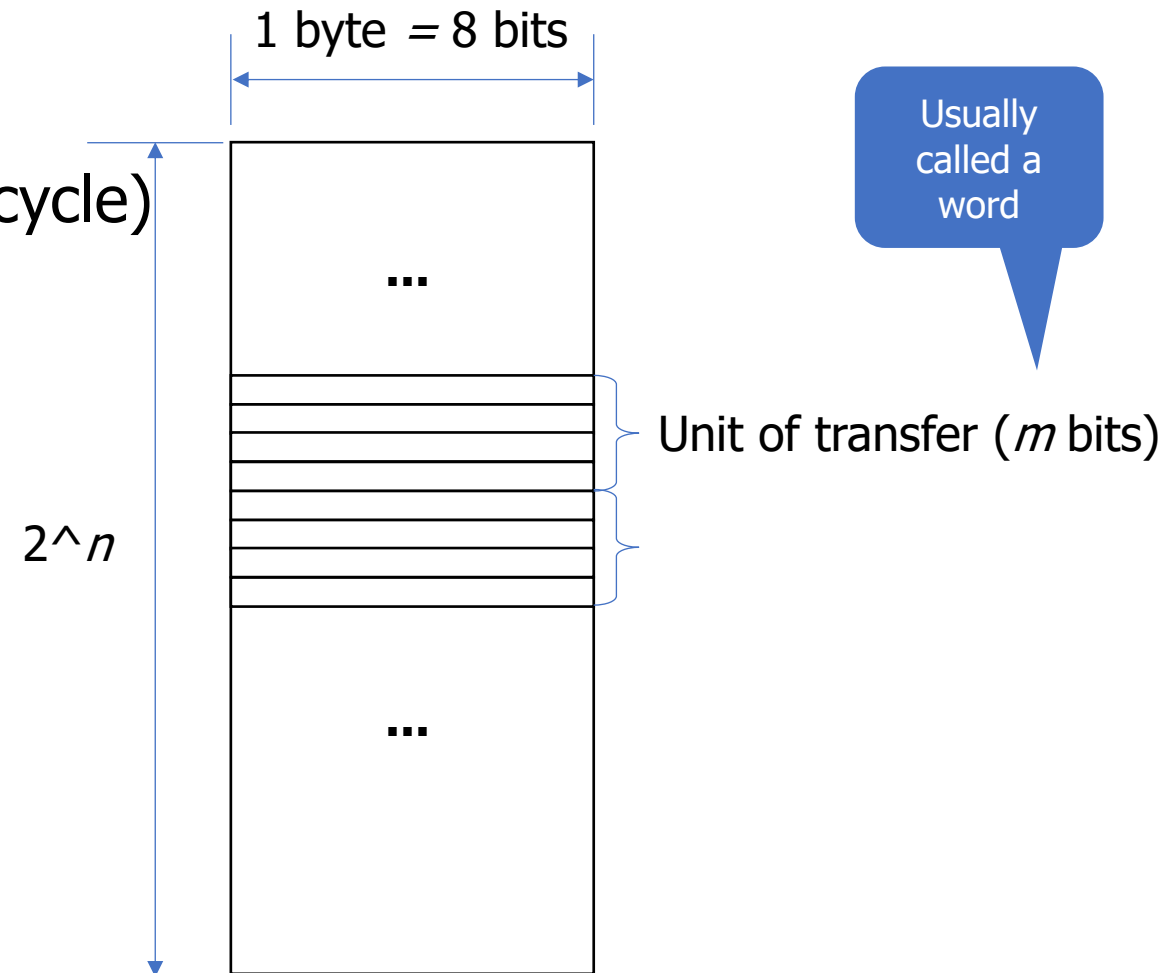


# System Bus & Address Space

- A virtual space what CPU sees through system bus
  - Assuming  $n$ -bit address bus and  $m$ -bit data bus ( $n = m$ , in most cases)
  - Addressable from 0 to  $2^n - 1$
  - At each address, one byte exist
  - $m$  bits can be transferred at once (each cycle)



[https://en.wikipedia.org/wiki/System\\_bus](https://en.wikipedia.org/wiki/System_bus)

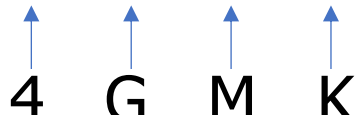


# Quiz

- Assuming n-bit address bus and m-bit data bus
  - Q: What is the size of the address space assuming n=32?

$$A: 2^2 \cdot 2^{10} \cdot 2^{10} \cdot 2^{10} = 4GB$$

4    G    M    K



Bus width

Bus speed

- Q: What is the maximum bus bandwidth assuming m=64 and a 100 MHz bus frequency?

A: on the next slide

# Max Memory Bandwidth = Bus Speed·Bus Width

Assuming 64-bit data bus

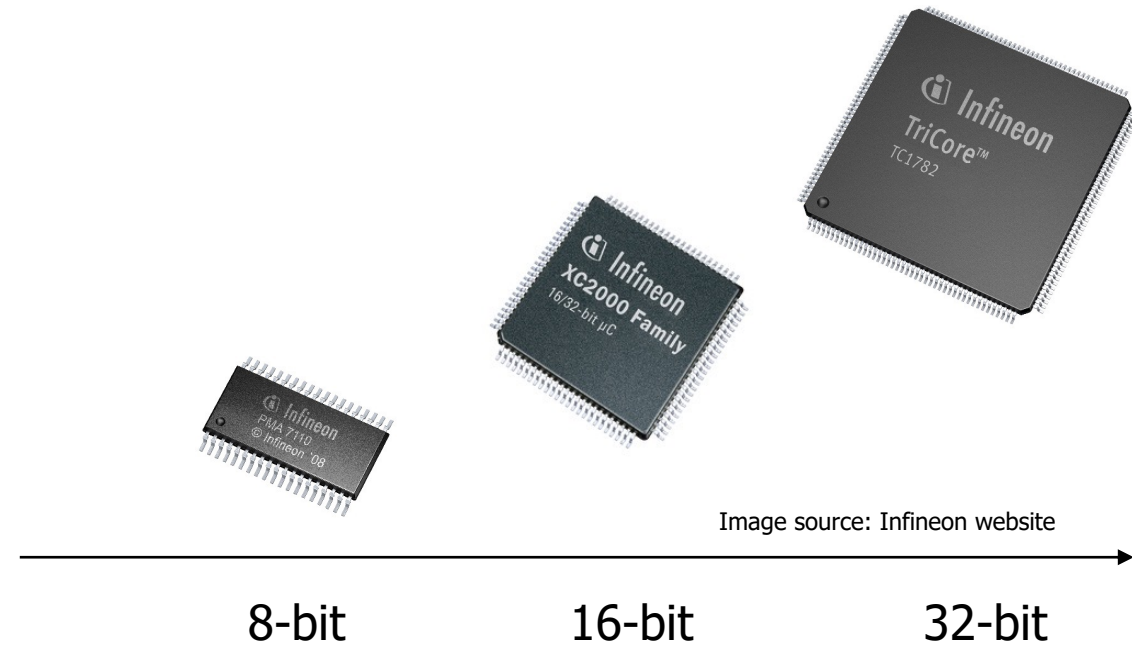
Table 1: SDRAM Memories Speed Comparison

Memory	Technology	Rated Clock	Real Clock	Maximum Transfer Rate
PC66	SDRAM	66 MHz	66 MHz	533 MB/s
PC100	SDRAM	100 MHz	100 MHz	800 MB/s
PC133	SDRAM	133 MHz	133 MHz	1,066 MB/s
DDR200	DDR-SDRAM	200 MHz	100 MHz	1,600 MB/s
DDR266	DDR-SDRAM	266 MHz	133 MHz	2,100 MB/s
DDR333	DDR-SDRAM	333 MHz	166 MHz	2,700 MB/s
DDR400	DDR-SDRAM	400 MHz	200 MHz	3,200 MB/s
DDR2-400	DDR2-SDRAM	400 MHz	200 MHz	3,200 MB/s
DDR2-533	DDR2-SDRAM	533 MHz	266 MHz	4,264 MB/s
DDR2-667	DDR2-SDRAM	667 MHz	333 MHz	5,336 MB/s
DDR2-800	DDR2-SDRAM	800 MHz	400 MHz	6,400 MB/s
DDR3-800	DDR3-SDRAM	800 MHz	400 MHz	6,400 MB/s
DDR3-1066	DDR3-SDRAM	1066 MHz	533 MHz	8,528 MB/s
DDR3-1333	DDR3-SDRAM	1333 MHz	666 MHz	10,664 MB/s
DDR3-1600	DDR3-SRAM	1600 MHz	800 MHz	12,800 MB/s

Source: <https://www.nxp.com/docs/en/supporting-information/BeyondBits2article17.pdf>

# 8-bit, 16-bit, 32-bit MCUs

- n-bit CPU
  - n-bit address bus
  - n-bit data bus
  - n-bit registers and ALUs
- With a larger n, what can we do?
  - Larger address space
    - Can handle larger memory
    - 32-bit CPU cannot handle memory > 4GB
  - More memory bandwidth
    - Linearly increases with n
  - Larger or more accurate numbers
    - 32-bit integer vs 64-bit integer
    - 32-bit float vs 64-bit double



# Address Space and Memory Size

- 32-bit system has 4 GB address space
- Do not confuse 4 GB *address space* with 4 GB *memory*



Pure abstract space for  
addressing

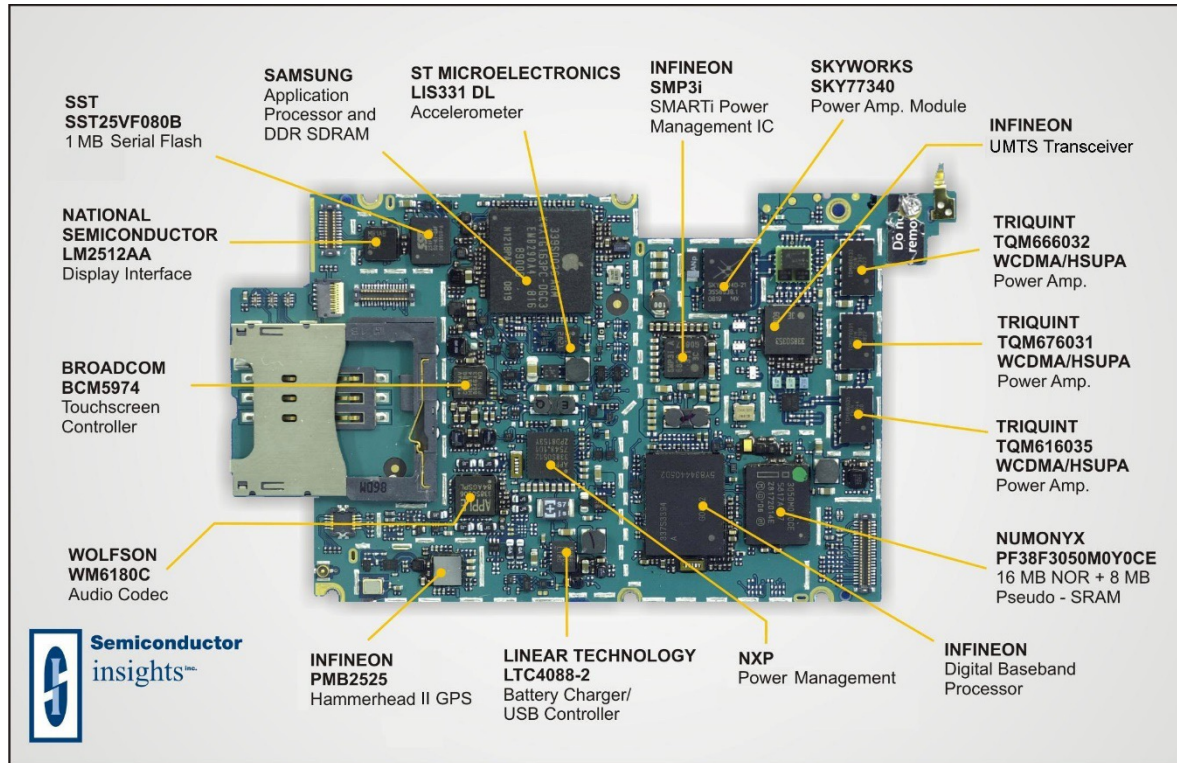
Physical component  
for storing bits and  
bytes

- What if 4 GB address space with 4 MB memory?
  - Only 4 MB of the 4 GB address space is used



# Memory Map

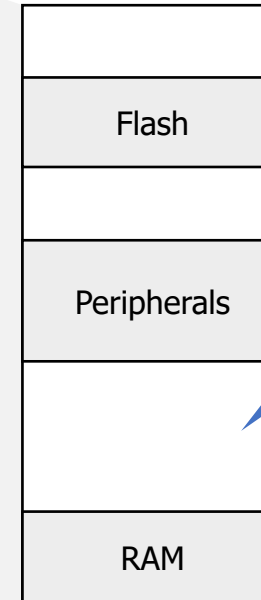
- Mapping of memory devices on an address space
- Peripherals (I/O devices) are also mapped to be accessed by CPU



[http://www.rapidrepair.com/guides/iphone3g/pmiPhone\\_boardtopBIG.jpg](http://www.rapidrepair.com/guides/iphone3g/pmiPhone_boardtopBIG.jpg)

Memory Map

High address

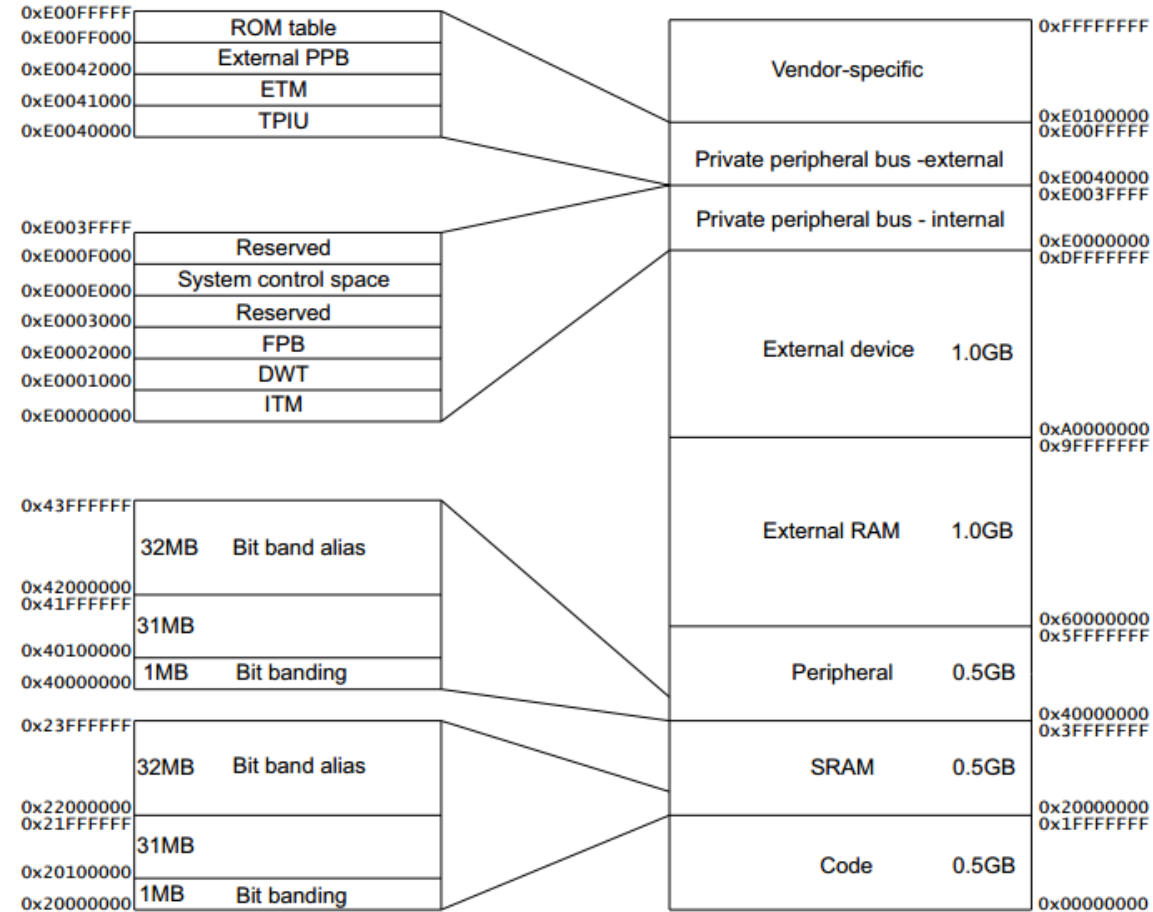


Low address

Q: What happens if we access an empty address?

# Example Memory Map

- ARM Cortex-M3 Memory Map



# Accessing Address Space (read)

```
void function(void)
{
    unsigned long p;
    p = *((unsigned long *)0x78787878);
}
```

Address



ARM compiler (32bit)

```
function():
    push    {r7}
    sub     sp, sp, #12
    add     r7, sp, #0
    mov     r3, #2021161080
    ldr     r3, [r3]
    str     r3, [r7, #4]
    nop
    adds    r7, r7, #12
    mov     sp, r7
    ldr     r7, [sp], #4
    bx      lr
```

0x78787878

# Accessing Address Space (write)

```
void function(void)
{
    *((unsigned long *) (0x78787878)) = 0x11111111;
}
```

Address

Data to be  
written



ARM compiler (32bit)

```
function():
    push    {r7}
    add     r7, sp, #0
    mov     r3, #2021161080
    mov     r2, #286331153
    str     r2, [r3]
    nop
    mov     sp, r7
    ldr     r7, [sp], #4
    bx      lr
```

0x78787878

0x11111111

# Summary

- Memory Technologies
  - RAM, ROM, Flash Memory
- Address Space
- Memory Map