Microprocessor systems

EMISY

Memory arrangement in microcontrollers Lecture 3

Semester 20L – Summer 2020
© Maciej Urbanski, MSc

email: M.Urbanski@elka.pw.edu.pl

WUT

ISE

ATTENTION – THIS IS NOT A DRILL

According to the dispositions of Polish Ministry of Science and Higher Education and the Warsaw University of Technology Headmaster, published on 11th March 2020, concerning suspension of all didactic classes in all universities in Poland from 12th March to 14th April

all EMISY course classes and activities are suspended to 14th April.

This will affect our lecture classes, project and lecture consultations, as well as laboratory classes.
All lecture materials for the suspension interval will be provided in Studia server. Consultations are available only via mail. I will inform you in case of any situation change.

Important remark

This material is intended to be used by the students during the Microprocessor Systems course for educational purposes only. The course is conducted in the Faculty of Electronics, Warsaw University of Technology.

The use of this material in any other purpose than education is prohibited.

This material has been prepared based on many sources, considered by the author as valueable, however it is possible that the material contains errors and misstatements.

The author takes no responsibility for the usage of this material and any potential losses this usage can lead to. Furthermore the author will be very grateful for pointing out any errors found and also for any other useful remarks on the course material and potential upgrades.

What should also be mentioned during last lecture

and what will be quite useful for the project...
Some pretty nice 8051 based microcontrollers and their manufacturers

Nice list, covering many 8051 microcontrollers can be found here:
http://www.keil.com/dd/chips/all/8051.htm

Examples:

Silicon Labs:
C8051F55x/56x/57x
EFM8 series microcontrollers – very nice

Microchip:
AT89LP51RD2-ED2-ID2

Cypress:
CY8C36 Family
Please make sure that MCU core is 8051 based!!

Image source: wikipedia.org

Silicon Laboratories EFM8SB1 block diagram

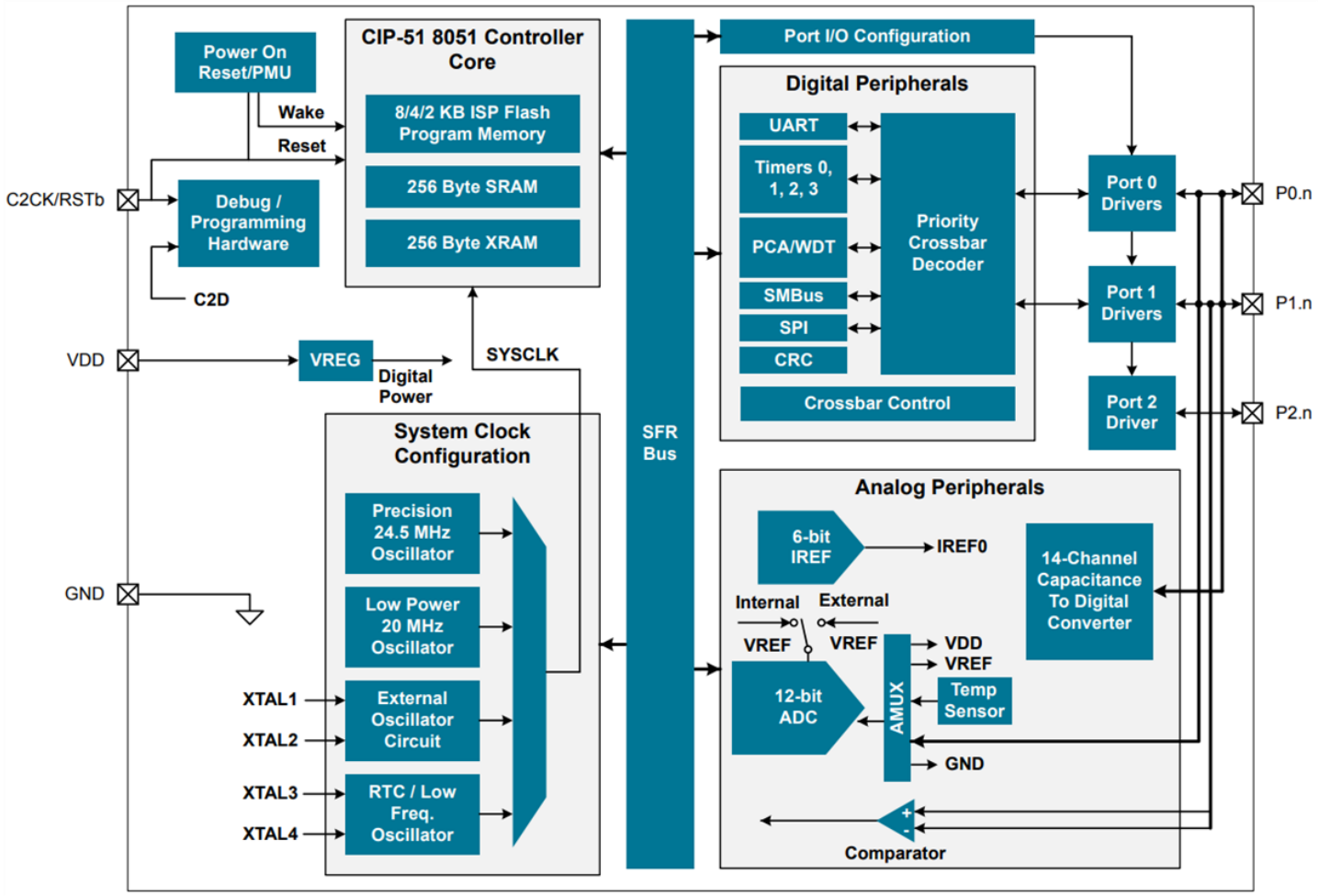
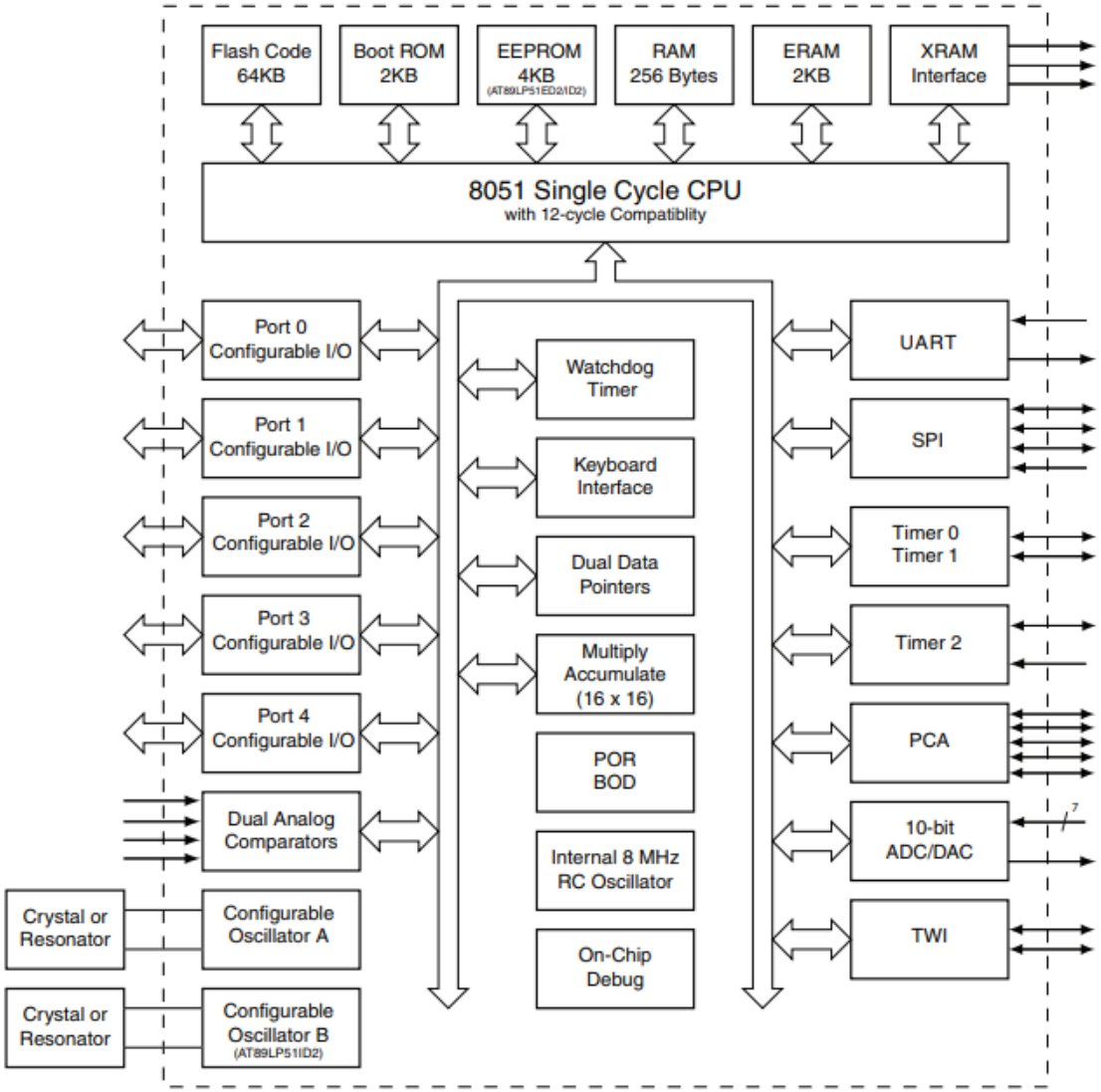


Image source: wikipedia.org

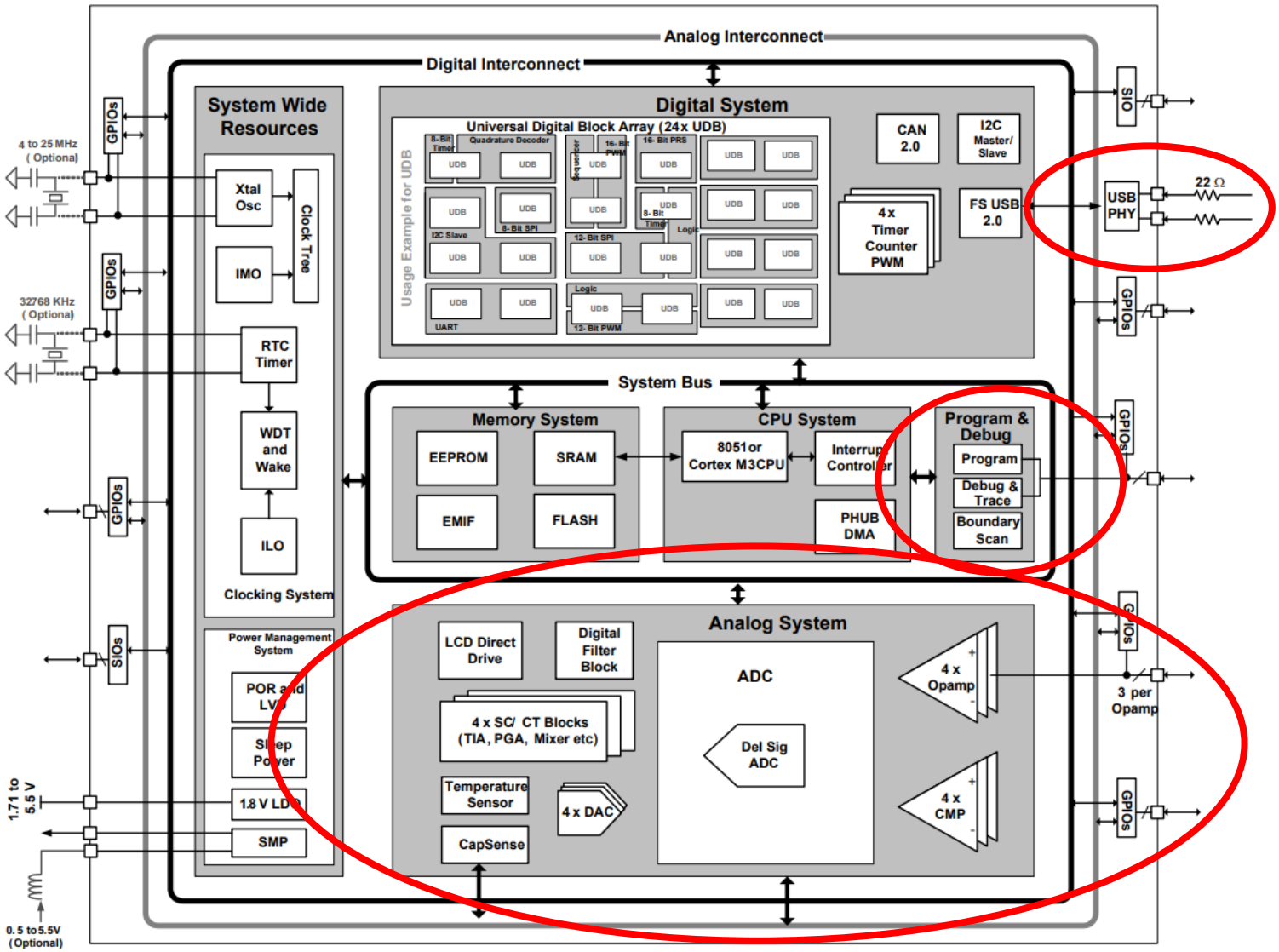
Microchip AT89LP51RD2-ED2-ID2

Figure 2-1. Atmel AT89LP51RD2-ED2-ID2 Block Diagram



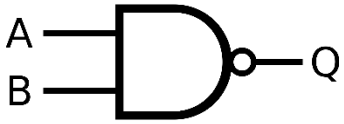
What should also be mentioned during last lecture

Cypress Semiconductor CY8C36 Family

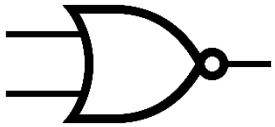


Logic gates and how to make synchronous logic circuits using them

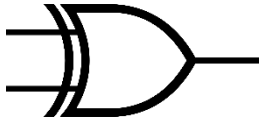
NAND gate (Not AND)

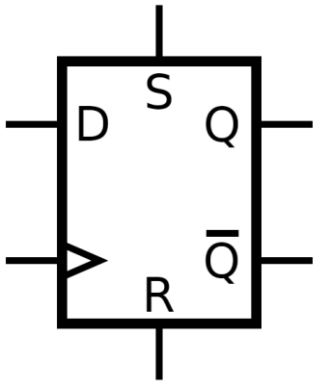


NOR gate (Not OR)

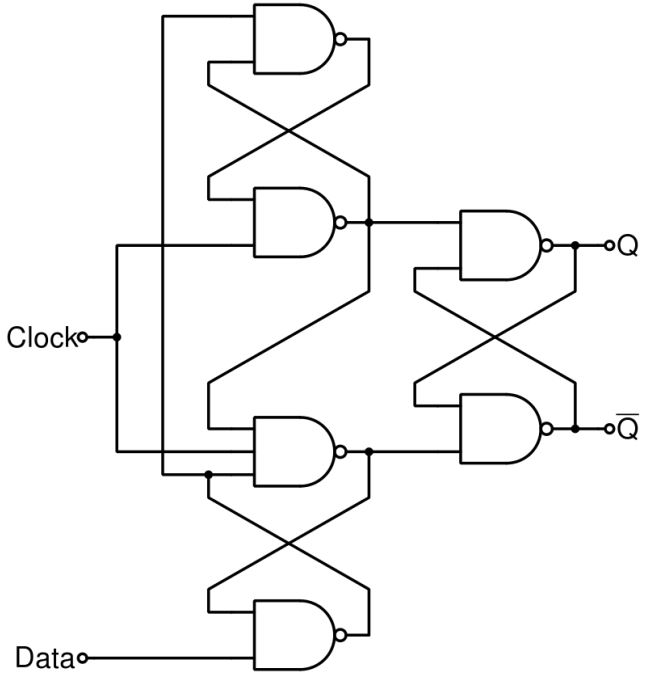


XOR gate (Exclusive O





D	Q(t)	Q(t+1)
0	0	0
0	1	0
1	0	1
1	1	1

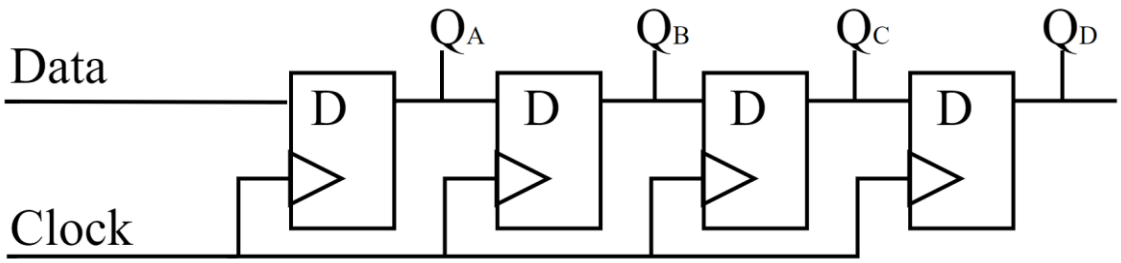


D type flip flop transfers D input state to Q output. Transfer takes place during CLK signal edge.

Image source: wikipedia.org

How to store information in digital circuits? Digital register types

The most basic type of memory is register. Serial register takes data from single input pin (serial input) and after n cycles outputs it (parallel output)



Parallel input register transfers the data to the output after single cycle.

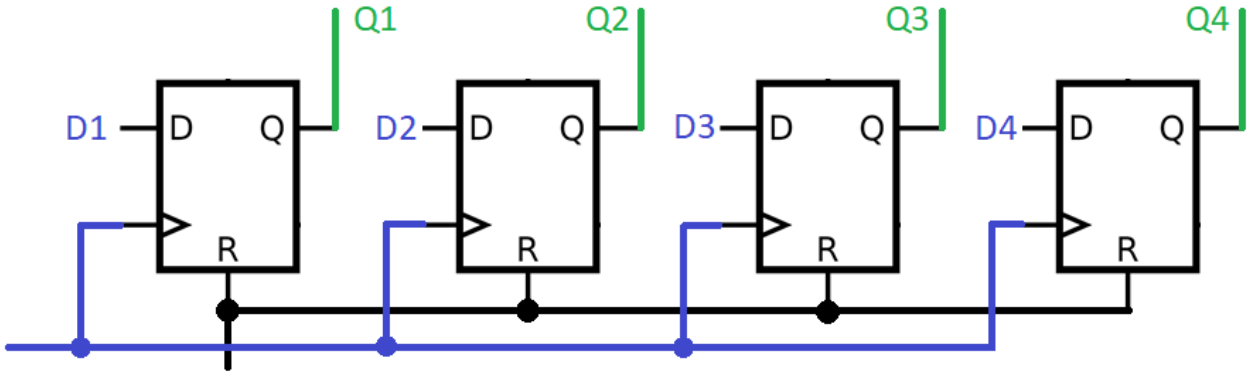


Image source: wikipedia.org

A basic block diagram of a simple RAM memory cell.

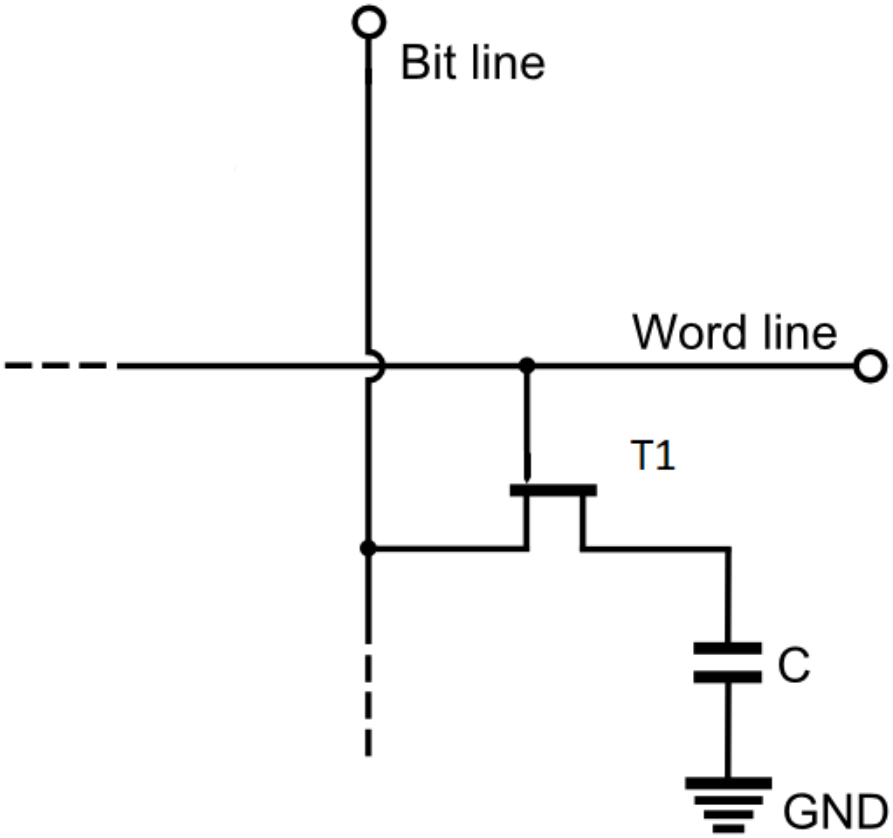
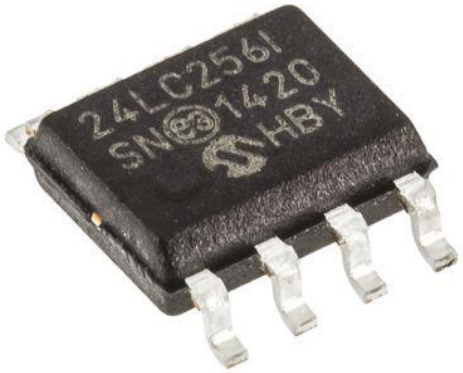


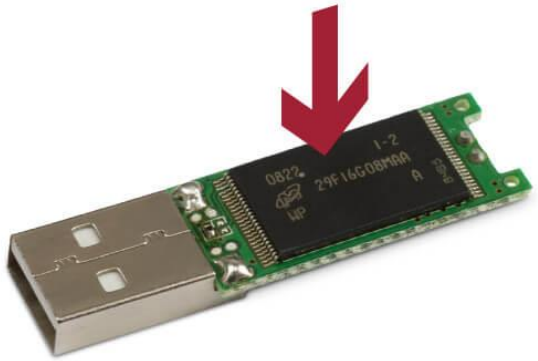
Image source: wikipedia.org

How to store information in digital circuits? Digital memory types

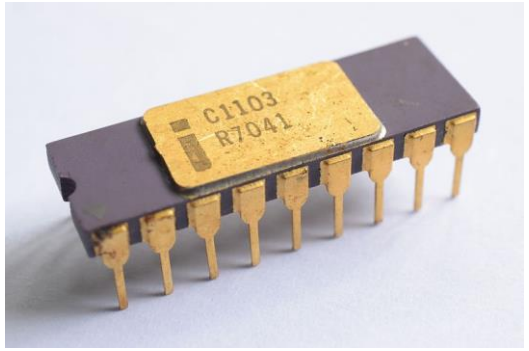
Jakie są typy pamięci – RAM, PROM, EPROM, EEPROM, FLASH, ROM



EEPROM I2C chip

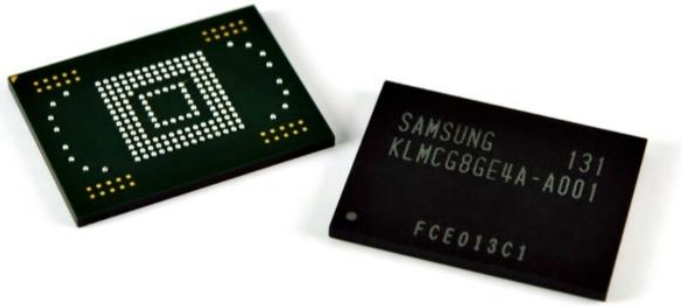


Flash memory
with USB interface



1970' Intel RAM memory chip

UV erasable EPROM memory



Modern RAM memory chips

How is information stored in digital memory? Binary data representation

Some say that the Matrix code is just a coded form of Japanese sushi recipe

The most widely used numeric system is decimal system.

Convenient, but to store it in digital memory 10 different logical states are needed.

That's why binary and hexadecimal numeric systems were introduced.

Binary system requires only two logical states.

Hexadecimal system is a convenient way to use binary numbers in code.

Decimal system representation

$$w = \sum_{i=0}^{n} 10^i a_i = a_0 + 10a_1 + \ ... + 10^n a_n$$

One hex symbol stores the same information as 4 binary symbols

Binary system representation

$$w = \sum_{i=0}^{n} 2^i a_i = a_0 + 2a_1 + \ ... + 2^n a_n$$

Hexadecimal system representation

$$w = \sum_{i=0}^{n} 16^i a_i = a_0 + 16a_1 + \ ... + 16^n a_n$$

Image source: wikipedia.org

How is information stored in digital memory? Binary data representation

Natural Binary System – stores information about unsigned numbers

$$67\ DEC = 2^6 + 2^1 + 2^0 = 64 + 2 + 1 = 01000011$$

8 bits can store decimals in range <0, 255>

U2 coding system – stores information about signed numbers.

To convert NBS unsigned to U2 signed number first negate all the bits of NBS number and then add 1. The opposite conversion works the same way.

$$67\ DEC = 2^6 + 2^1 + 2^0 = 64 + 2 + 1 = \quad 01000011$$
$$-67\ DEC = \sim(2^6 + 2^1 + 2^0) + 1 \quad = 10111100 + 1 = 10111101$$

$$-2\ DEC = \sim(2^1) + 1 = \sim(00000010) + 1 = 11111110$$

During conversion do not forget about leading zeros!!

8 bits can store decimals in range < -127, +127>

BCD coding system – used in LED 7 segment driver chips

Each digit is represented with 4 bits, codes as unsigned decimal (NBS)

$$6\ DEC = 0110\ BIN, \qquad 7\ DEC = 0111 \Rightarrow 67\ DEC = 01100111\ BIN\ BCD$$

Image source: wikipedia.org

Memory arrangement in MCUs – 8051 example

Intel 8051 based microcontrollers can have up to 64 kB of program memory.

They allow to connect external memory.

Memory is divided into program memory and data memory.

It is possible to lock the ability of reading the microcontroller memory – protection against reverse engineering.

There is a special set of registers in memory – so called SFRs or Special Function Registers.

On the other hand some more modern microcontrollers have bigger internal memory – STM32F407VGT6 (F4 Discovery) has 1 MB of internal FLASH memory – 8 Times more than the biggest 8051 with the biggest external memory...

What to choose then?

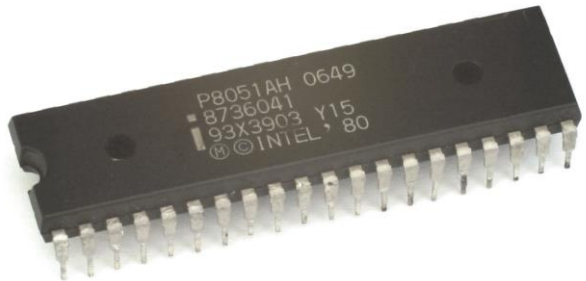


Image source: wikipedia.org

Intel 8051 memory map

Memory map describes assignment of individual memory and I/O resources to the microcontroller address space.

It helps in further hardware and firmware development. It is particularly useful in case of microprocessor systems containing many memory chips and I/O devices.

First three bytes of program memory of 8051 is for the initialization. Registers from 3 to 42 are reserved for interrupts.

Data memory of the 8051 – 256 bytes of RAM and 128 bytes of SFRs.

The bottom section of the 8051 data memory consists of 4 register banks – RB0 to RB3 – these are general purpose registers., used to store partial results, constants, etc.

Next (20H-2FH) there are 16 bytes that can be bit addressed – 128 directly addresable bits.

128 bytes in the internal data memory are reserved for Special Function Registers – SFRs. They are overlapped with 128 bytes of upper RAM – the proper addressing mode has to be chosen for accessing SFRs or upper RAM.

Memory between 30H and 7FH can be used for general purposes.

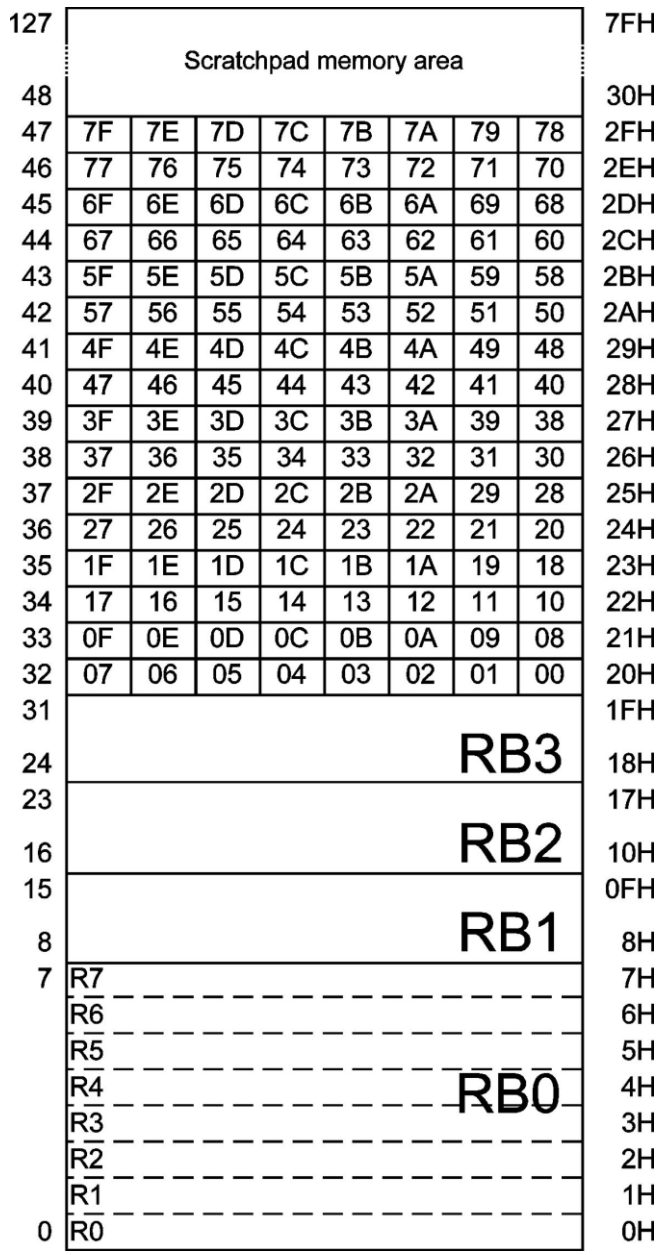


Image source: wikipedia.org

Intel 8051 memory map – special function registers

Internal data memory, between addresses 128 and 255, is reserved for special function registers. They are used to control the operation of microcontroller. All the operations between the CPU and internal peripherals are done using the SFRs. They can be addressed directly **only**.
Unused SFRs can be used as general purpose registers.

Some of the SFR registers:

A – accumulator
B - register used in multiplication and division operations
PSW – program status word
SP - stack pointer
DPH, DPL – data pointer high, low – 16 bit DPTR
IPC - interrupt priority control
IEC - interrupt enable control
TMOD - timer mode
TCON - timer control
TH0, TL0 - timer 1 high, low
TH1, TL1 - timer 2 high, low
SCON, SBUF – serial control and serial buffer

F8									FF
F0	B 00000000								F7
E8									EF
E0	ACC 00000000								E7
D8									DF
D0	PSW 00000000								D7
C8									CF
C0									C7
B8	IP xxx00000								BF
B0	P3 11111111								B7
A8	IE 0xx00000								AF
A0	P2 11111111								A7
98	SCON 00000000	SBUF xxxxxxxx							9F
90	P1 11111111								97
88	TCON 00000000	TMOD 00000000	TL0 00000000	TL1 00000000	TH0 00000000	TH1 00000000	(AUXR) xxxxxxx0		8F
80	P0 11111111	SP 00000111	DPL 00000000	DPH 00000000				PCON 0xxx0000	87

Image source: wikipedia.org

Special function registers of 8051 – accumulator and stack

Accumulator is one of the most important special function registers. It usually stores one of the instruction arguments, or arithmetical or logical results of operations.

Without it every partial result would have to be stored in the main memory, which would be significantly slower that writing it to SFR accumulator.

The lenght of the accumulator limits the word lenght of the microcontroller.

In 8051 there is only one register working as the accumulator.
In AVR microcontrollers each general purpose register can be used as an accumulator.

Stack is a type of memory that gives the access only to the last element . Information is written to the stack (placed on top of the stack) and the address of the top element is stored in stack pointer (SFR). Stack is placed in the microcontroller data memory. It is up to the designer of the firmware to place the stack in memory in a way that prevents overwriting of the memory.
It is LIFO type memory that is used to store jump return addresses (subroutines)

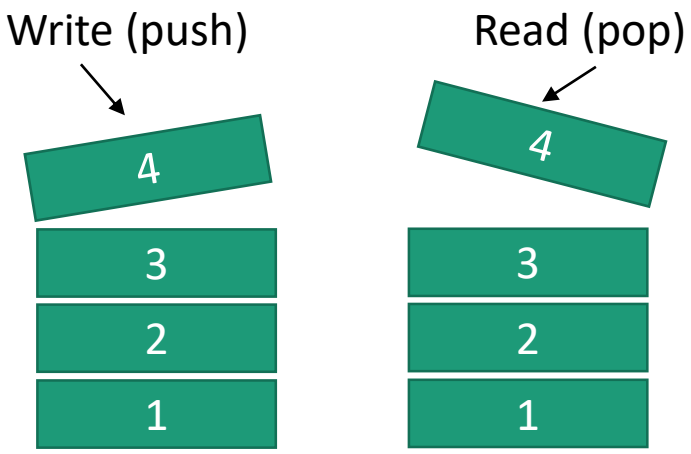
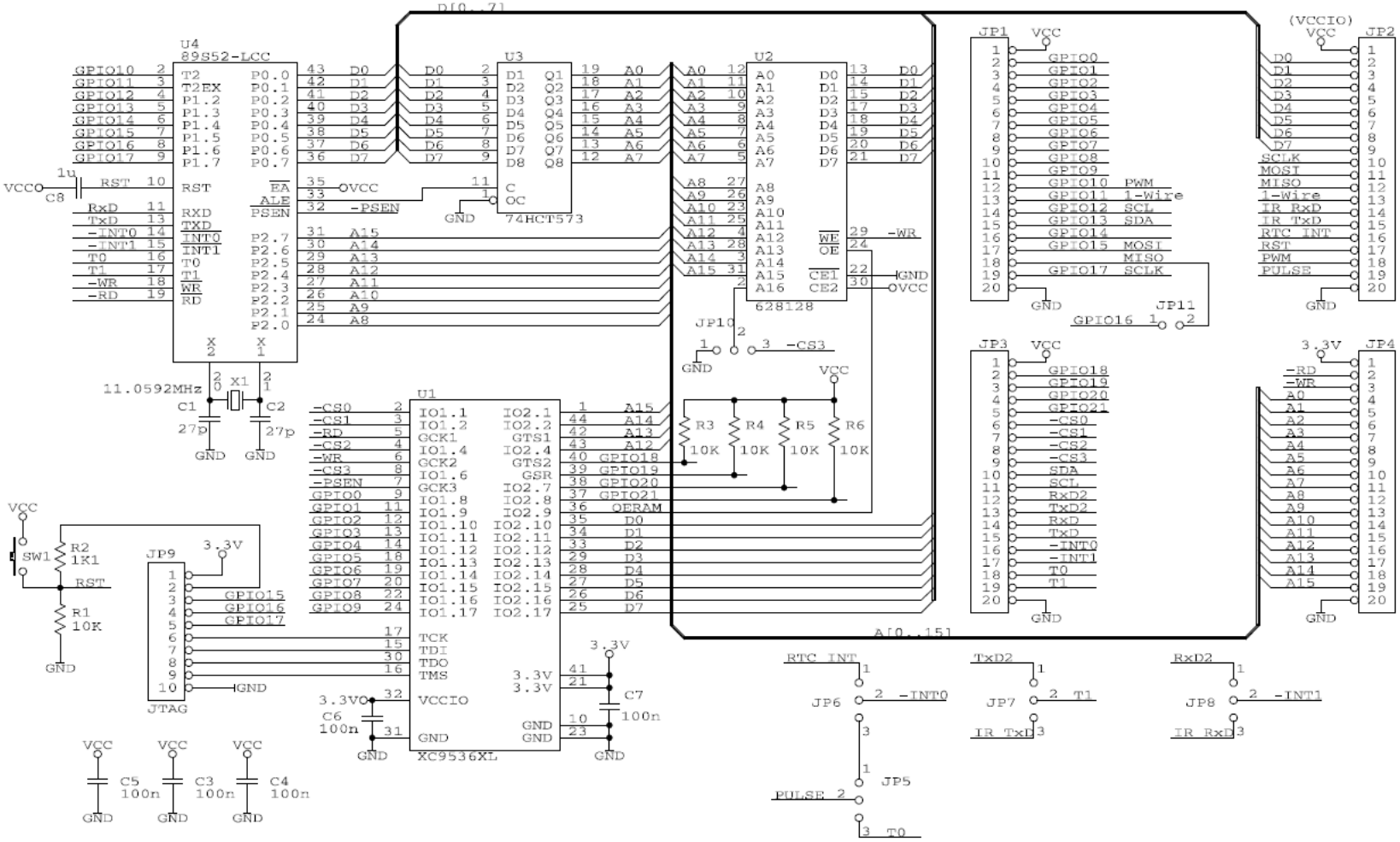


Image source: wikipedia.org

External memory - interfacing



Laboratory kit microprocessor system electrical schematic.

External memory - interfacing

Almost all 8051 microcontrollers can use external program memory. The program memory selection (internal/external) is done based on EA pin state.

To use internal program memory the EA pin has to be logic high.

To use external program memory the EA pin has to be logic low.

The upper address byte is sent via P2, the lower via P0.
ALE signal is used to latch the external memory data byte. It drives the external latch (74573 type).

PSEN signal is used to read from the program memory via P0.
RD signal is used to read the data from external data memory via P0.
WR signal is used to write data to external data memory via P0.

More information https://www.mikroe.com/ebooks/architecture-and-programming-of-8051-mcus/memory-organization

Instruction list

Instruction is an elementary operation that microcontroller can perform. Instructions represent simple operations, using the data stored in memory and registers.

The set of all instructions that the microcontroller can execute is called the instruction list. This list should be functionally finite, which means that it should allow to design any algorithm.

Each instruction is stored in memory as coded string of zeros and ones. The code of one instruction can take one or more memory cells.

The designer is writing the code using mnemonics – symbolic names for registers and instructions. The code is then assembled by a special program, called assembler.

Arguments of instructions are stored in memory. Addresses in the memory that point to the arguments can be defined in several ways, called addressing modes.

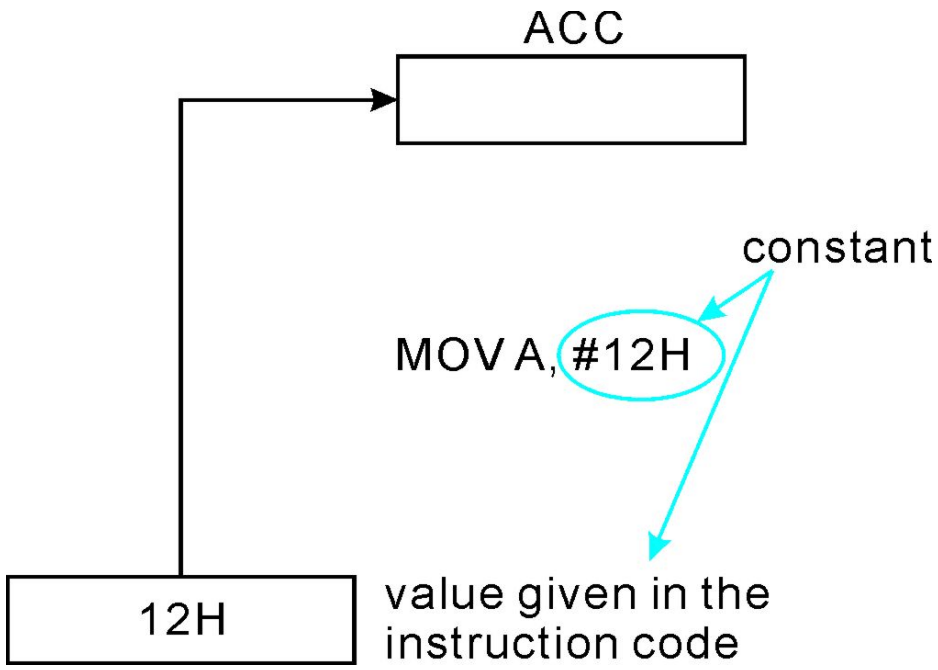
Code presented to the right is a simple P3.0 LED blinker, written for 8051 microcontroller (AT89S4051).

Much more sophisticated assembler code will have to be developed during laboratory classes...

```
start:  mov   P3, #0x00
l1:     mov   r1, #0xff
        mov   r0, #0xff
l2:     djnz  r0, l2
        djnz  r1, l1
        mov   P3, #0x02
l3:     mov   r1, #0xff
        mov   r0, #0xff
l4:     djnz  r0, l4
        djnz  r1, l3
        sjmp  start
              end
```

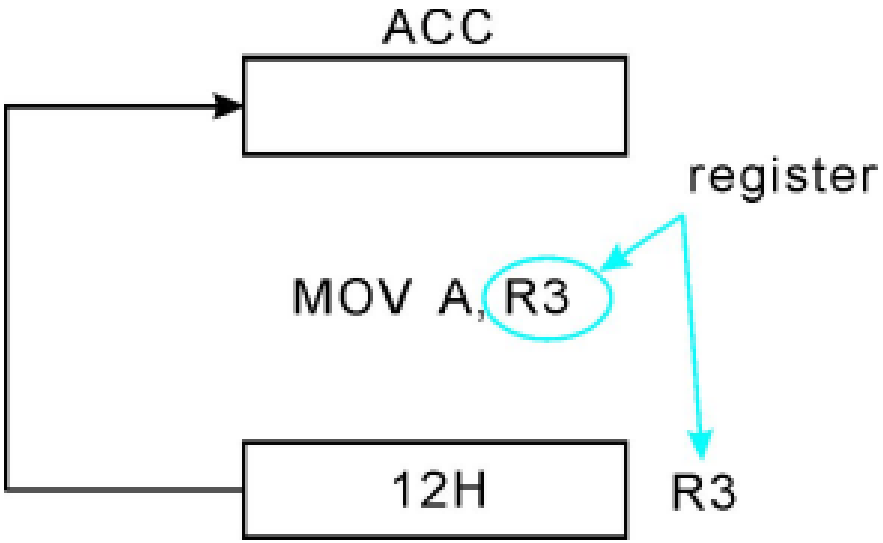
Addressing modes – immediate addressing

In immediate addressing the argument is placed in memory after the operation code. It is most commonly used for addressing the constants.



Addressing modes – register addressing

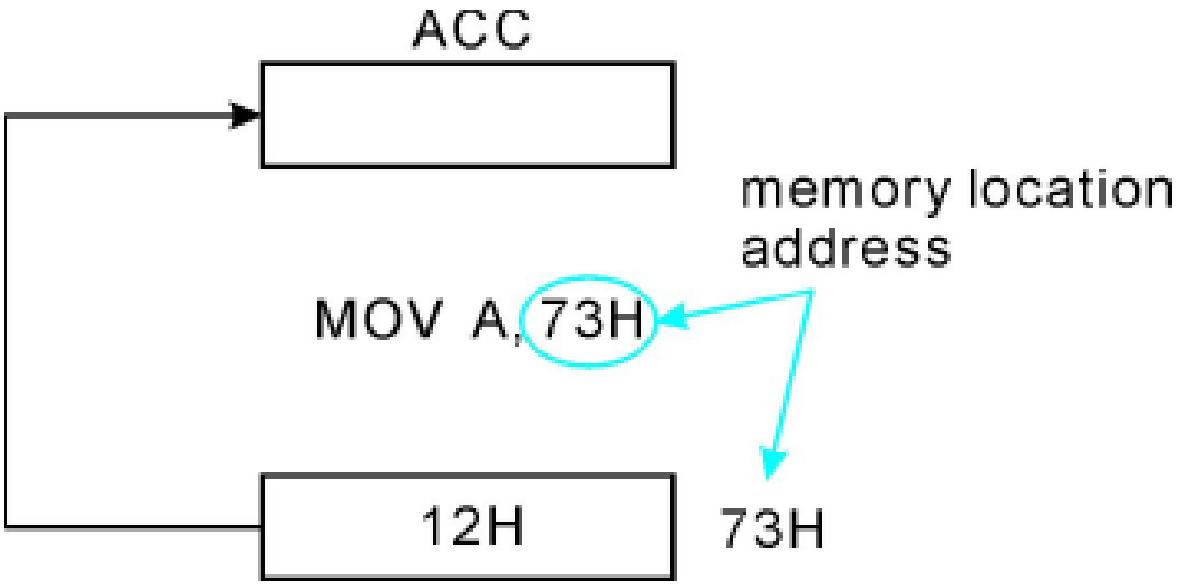
Register addressing allows access to general purpose registers in currently selected register bank. This selection is done by the PSW register.



In register addressing the argument's address is placed in one of the microcontroller's registers. It points at the place in memory, where the data is stored. In the example the value 12H will be moved to the accumulator.

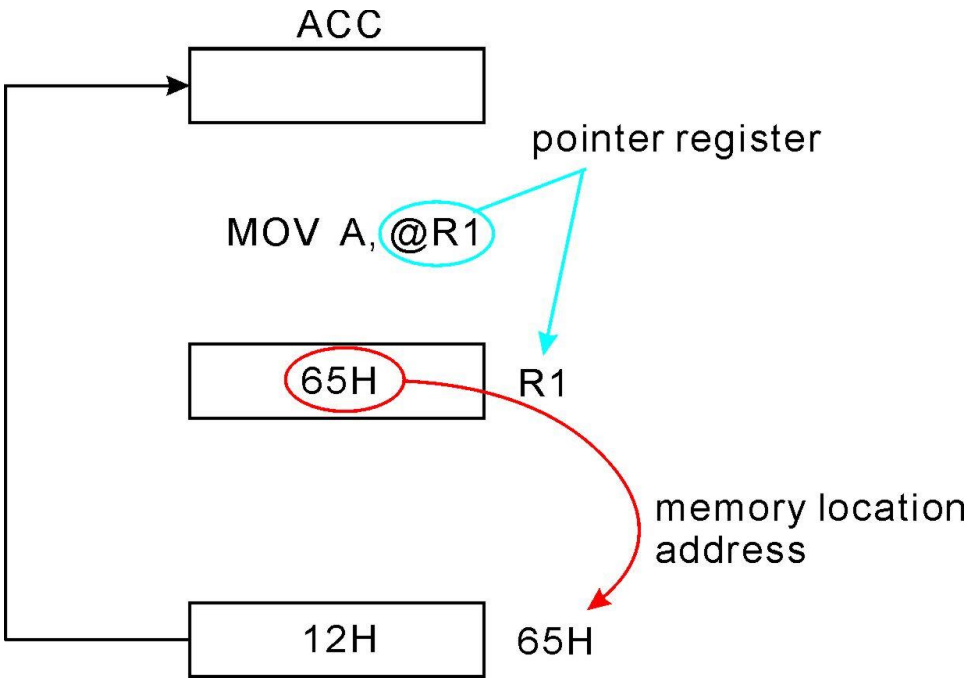
Addressing modes – direct addressing

In direct addressing mode the argument address is placed in memory as a part of instruction code, directly after the instruction code (like in immediate addressing).



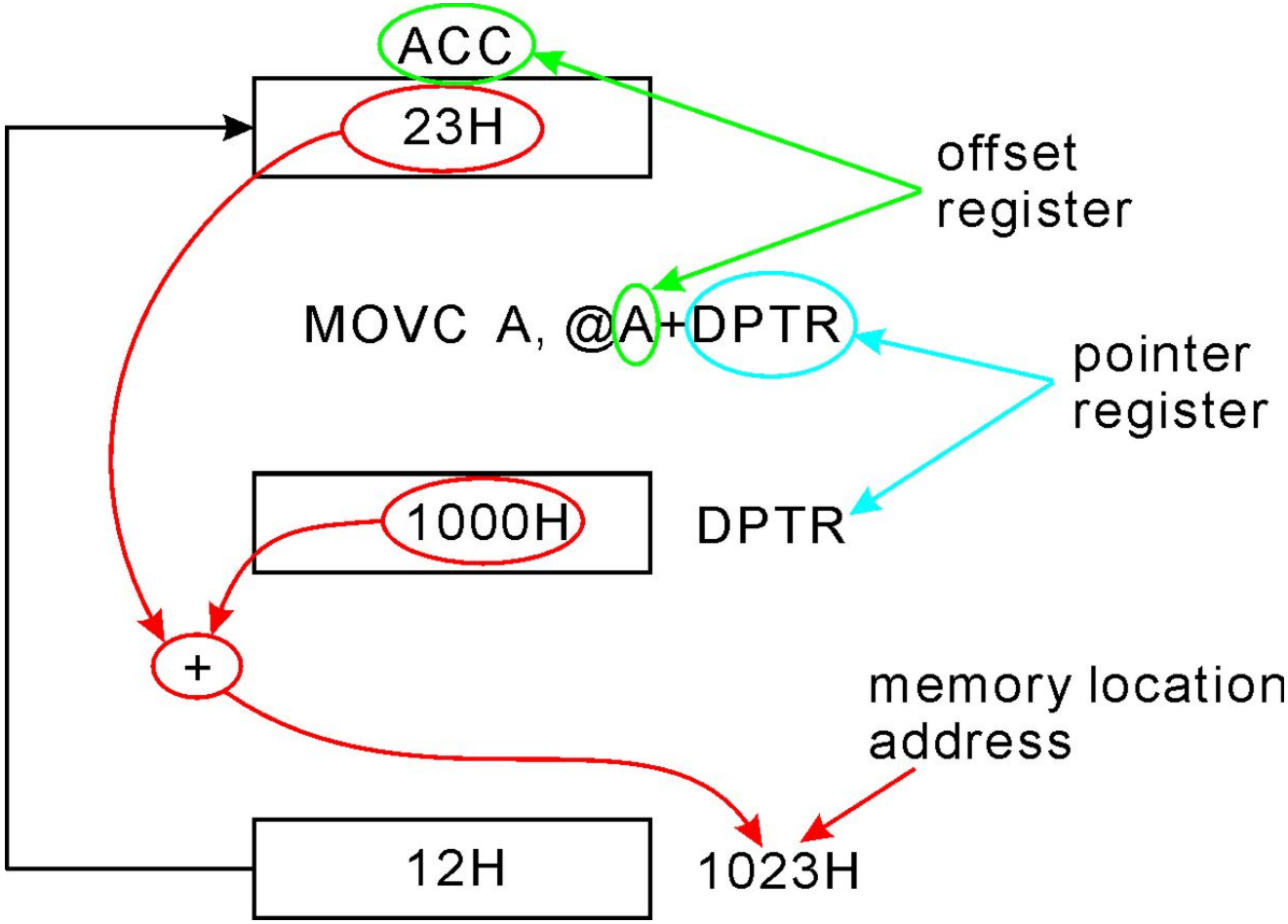
Addressing modes – indirect addressing

In indirect addressing, a register that stores the address of another register is specified in instruction. It is used for getting the information from available RAM locations. It uses R0, R1 or stack pointer for specifying the address. It allows to access the internal RAM only.



Addressing modes – indirect addressing with offset

The other name is indexed addressing. It uses a base register in forming the final address for jump instructions. It is used for jump tables or lookup tables.



Other types of memories in microprocessor systems

EPROM and EEPROM memory – non-volatile memory types

EPROM – Erasable PROM – a type of memory, usually available as an external chip that should be embedded into the microprocessor system. Its cell is based on a custom MOS transistor with two gates – one for writing and one for reading.

Older EPROM chips were UV erasable – this means that they had to be placed in special UV erasers to wipe out their contents.

Nowadays they are obsolete and replaced with FLASH and EEPROM memories.

EEPROM – Electrically Erasable PROM – a type of memory, available both as internal microcontroller memory peripheral and as a standalone chip. It can be erased using electrical signals.
It is non-volatile – its contents is not wiped out during power off.
Usually driven using I2C or SPI interfaces.
It can withstand a limited number of write cycles – usually about 10000.
It is used for example to store configuration information in microprocessor systems.

Image source: en.wikipedia.org

Other types of memories in microprocessor systems

SRAM memory – volatile memory

SRAM – Static Random Access Memory – is a semiconductor type of memory used in microprocessor systems. Static type means that its contents is not damaged as long as the power is on. Each bit of the memory is organized as a 4 transistor flip flop, driven via additional two transistors. Unlike DRAM, such structure does not require refresh (information is stored in the flip-flop, not the capacitance).