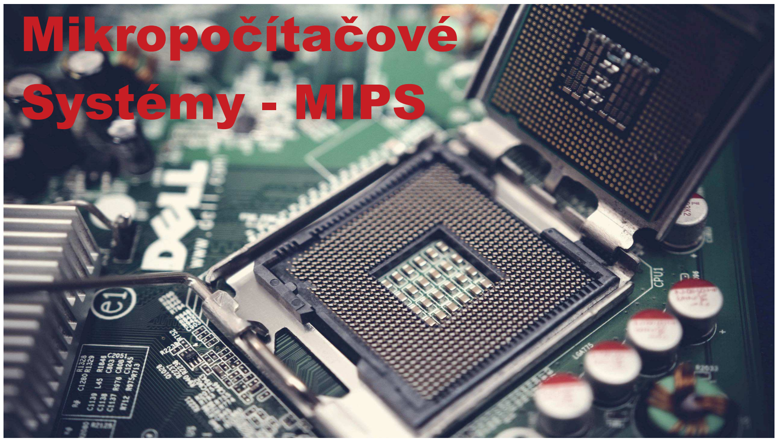
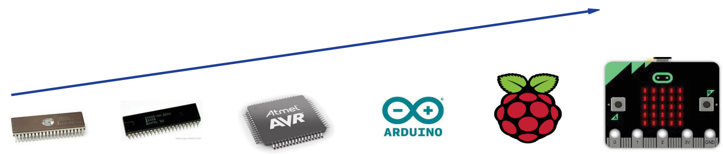
Mikropočítačové Systémy - MIPS



Richard Balogh
Štefan Chamraz

STU FEI

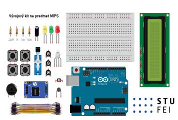


8048 | 8051 | Atmel AVR | Atmel AVR | Raspberry Pi | micro:bit

Cvičenia

Google Classroom

senzor

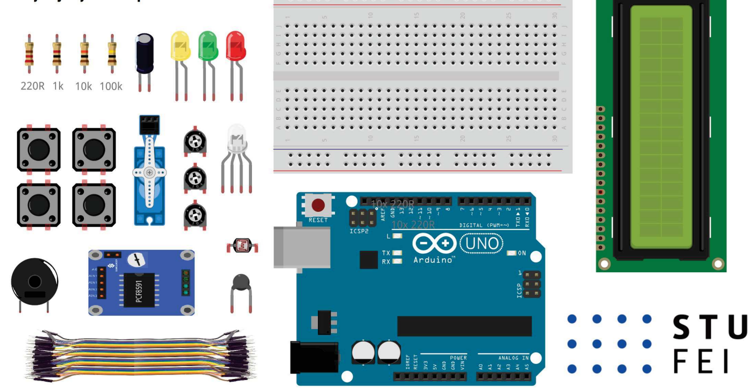


Podmienky na absolvovanie predmetu:

Cvičenia: X bodov za úlohy z cvičení (2021: 43b.) **TERMÍN?**
a záverečný projekt (20b.)

Skúška : 100 - X bodov (2021: 37b)

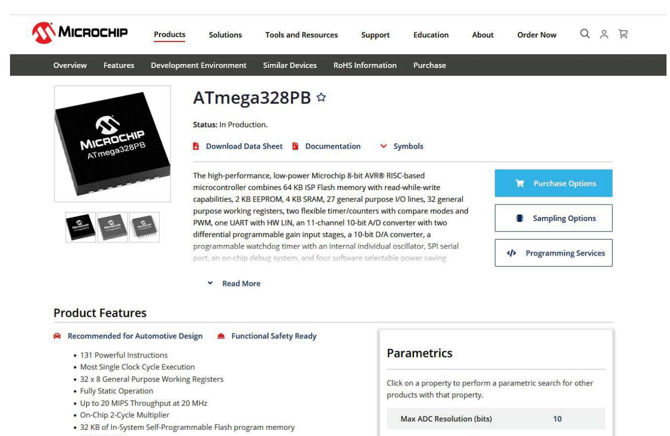
Vývojový kit na predmet MIPS



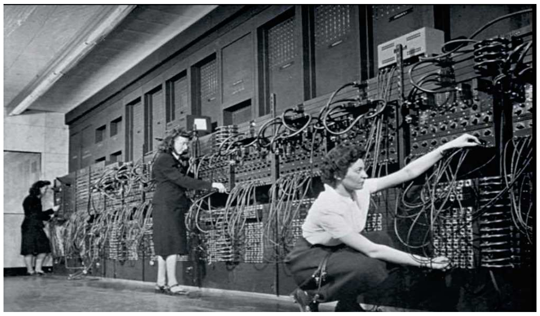
profesia

arduino

embedded c

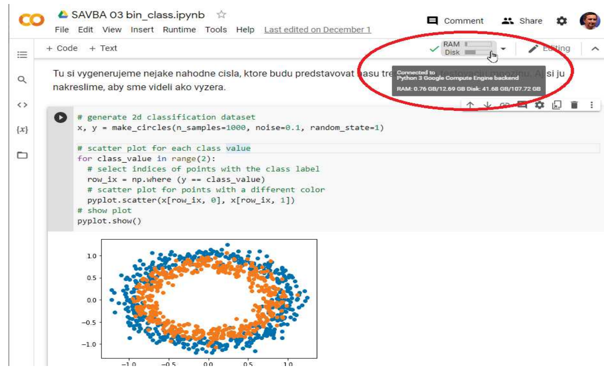


Physical / Virtual Computing



Jean Jennings, Marlyn Wescoff a Ruth Lichterman programujú počítač ENIAC na Pennsylvánskej univerzite v USA, 1946.

Physical / Virtual Computing



Vývoj HW

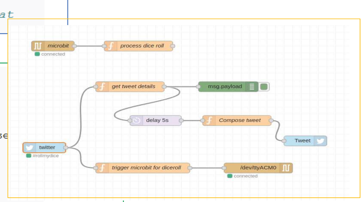
4+1		4+4
1 MB pamäte	6000 x	6 GB RAM + 64 GB sd
550 W	100 x	5 W
29 kg	150 x	191 g
1,2 MIPS	8000 x	10 000 MIPS

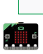
Vývoj SW

```
              .ORG     0030h
              DB       03h,01h,02h,03h
              .ORG     0000

              LDA      0030H
              MOV      C,A ; Initialize C-counter
              SUB      A ; sum = 0
              LXI      H,0031H ; Initialize pointe
BACK:         ADD      M ; SUM = SUM + data
              INX      H ; increment pointer
              DCR      C ; Decrement counter
              JNZ      BACK ; if counter 0 repeat
              STA      040H ; Store sum
```

```
 for (;;)
 {
  c = (char)rand();
  printf("%d,%d\r",(int)(100*(sin(x)+1)),c);
  if (c>245)
    printf("Alarm! Temperature HIGH!!!\r");
  if (c<5)
    printf("Alarm! Temperature LOW!!!\r");
  _delay_ms(300);
  x=x+0.1;
  }
 return 0;
} /* End Of Main */
```

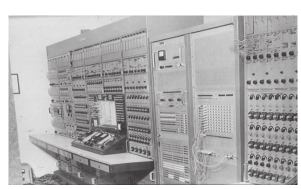
```
from microbit import *

display.show(Image.YES)

while True:
    if button_a.is_pressed() and button_b.is_presse
        display.show(Image.HEART)
        sleep(500)
    elif button_a.is_pressed():
        display.show(Image.HAPPY)
    elif button_b.is_pressed():
        display.show(Image.SAD)
```

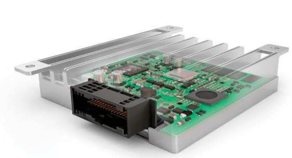




Vývoj programovacích jazykov: Asembler, jazyk C, Python a Node-Red

Miniaturizácia





Miniaturizácia

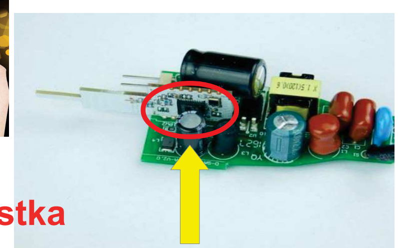
Moorov zákon

Ak by sa rovnako rozvíjal automobilový priemysel:

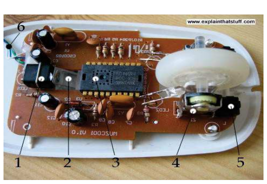
300 000 km/h **850 000 km / liter** **0,04 $ / ks**

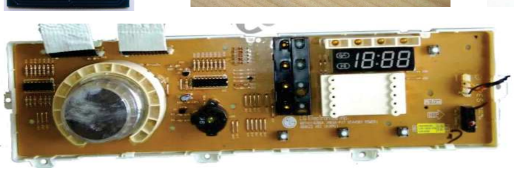
Embedded Computing

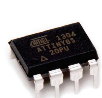
Embedded Computing / vnorené systémy



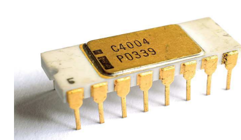
počítač = súčiastka

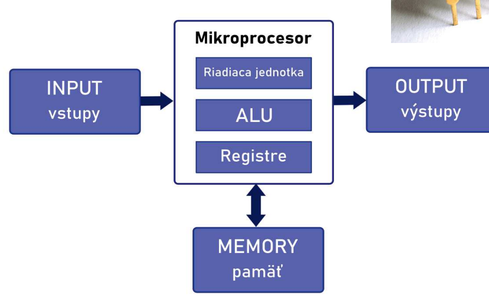






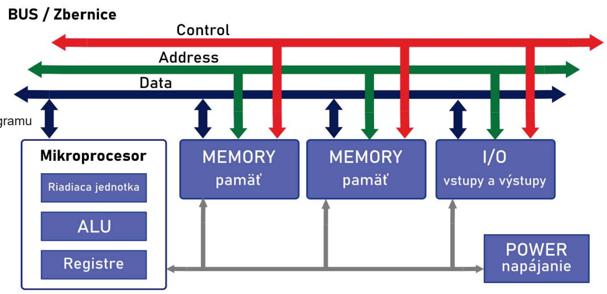
Mikroprocesor



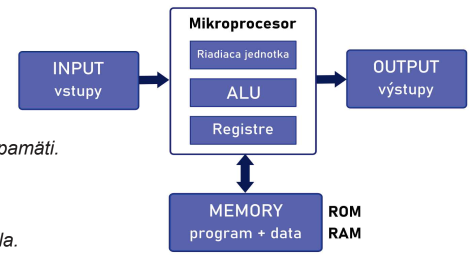


Štruktúra počítača

- Procesor
  riadi, realizuje výpočty
- Pamäť
  pamäť dát & pamäť programu
- Input/Output
  interface na okolitý svet
- Napájanie



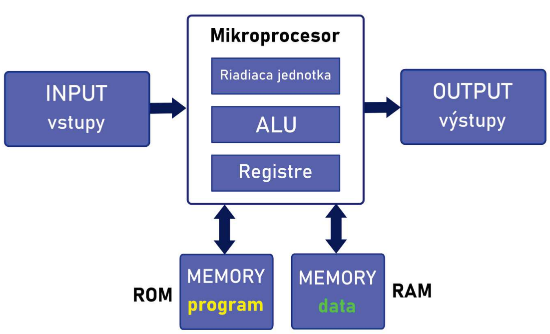
Architektúra Von Neumann (*Princetonská)*



Program a Dáta: *sú uložené v pamäti.*
- Inštrukcie,
- dáta (čísla, znaky, ...),
- adresy,
sú ukladané ako dvojkové čísla.

Požadujeme: *rýchly procesor a veľkú pamäť* =>
Presúvanie informácií: *„von Neumannov bottleneck" – John Backus 1977.*

Harvardská architektúra (Aiken)

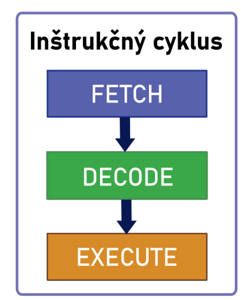


Inštrukčný cyklus

- Výber inštrukcie z pamäte

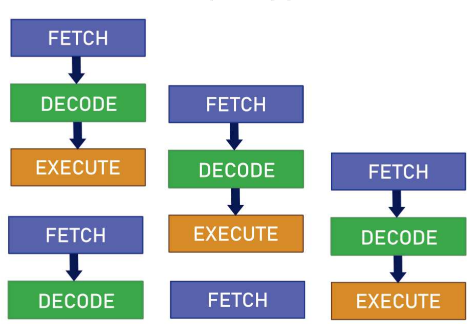
- Dekódovanie inštrukcie
- Výber operandov

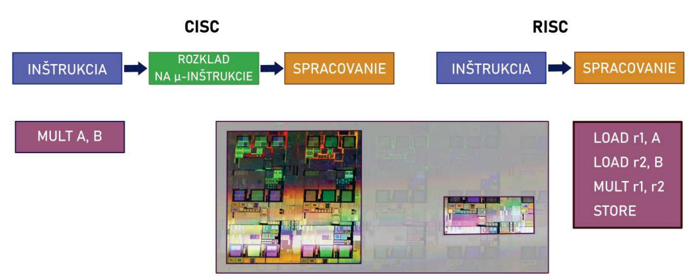
- Vykonanie požadovanej operácie
- Zápis výsledkov do pamäte



Paralelizmus: pipeline

3-stupňová pipeline





CISC: *málo* (4-100)

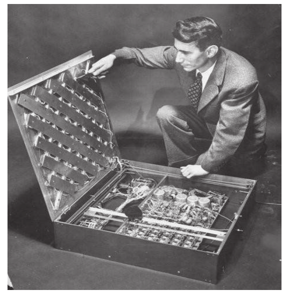
execution time = počet inštrukcií x CPI x frekvencia

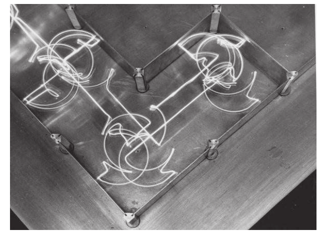
RISC: *veľa* 1

Historické okienko

1952

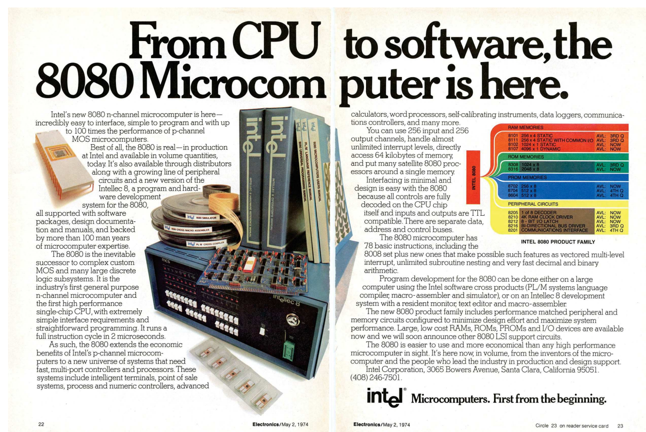
C. Shannon - Theseus:
Maze-Solving Mouse (1952)





Source: http://cyberneticzoo.com

1974



Intel 8080

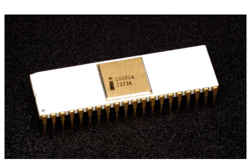
Produced April 1974; 43 years ago(!)

CPU clock:	**2 MHz to 3.125 MHz (8080-A)**
Package:	**40-pin DIP**
Power:	**+5, -5 and +12 V, ~ 1,3 Watt**
Military	**– 55 ºC to +125 ºC**
Technology:	**6 µm, 6 000 transistors, die size 20 mm²**

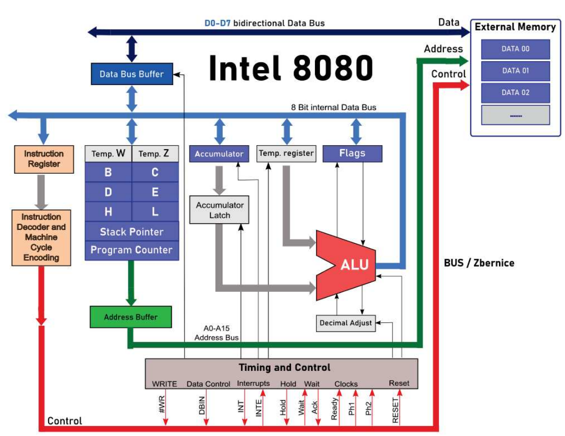
Common manufacturer(s)

Intel, AMD, NEC, National Semiconductor, OKI, Siemens, Texas Instruments, Sovietsky zväz KR580VM80A, Poľsko MCY7880, ČSSR Tesla Piešťany MHB8080A

r. 1979 – 500 000 ks / mesačne za cca $3 – 4 / ks



By Konstantin Lanzet - CPU collectionCamera
https://commons.wikimedia.org/w/index.php?curid=7028099



Inštrukčná sada procesora 8080

• Skladá sa z
– 74 základných inštrukcií, napr. MOV
– 246 konkrétnych inštrukcií, napr. MOV A,B

• 8o8o inštrukcie delíme na

1. **Data Transfer (Copy)**	**Presunové**
2. **Arithmetic**	**Aritmetické**
3. **Logical and Bit manipulation**	**Logické**
4. **Branch**	**Skokové**
5. **Machine Control**	**Riadiace**

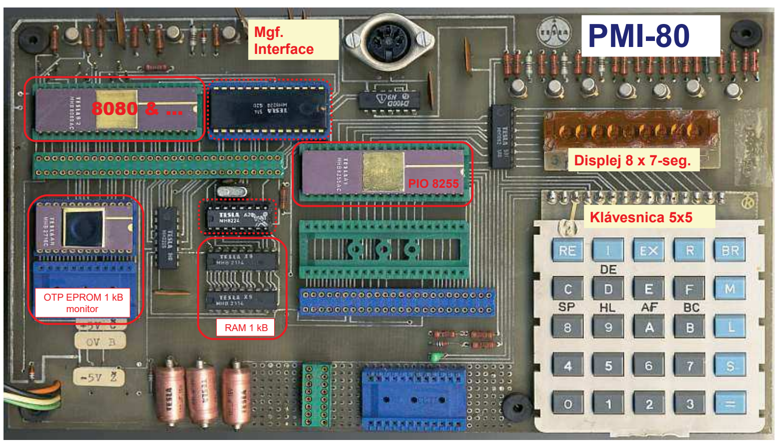
Strojový kód vs. jazyk symbolických adries

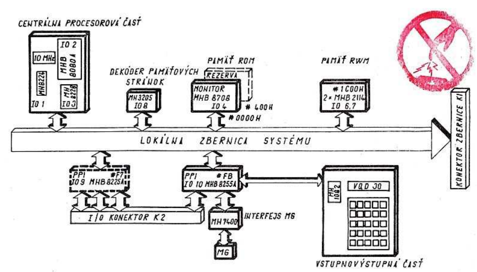
```
       ; sucet dvoch cisel A a B
START: MVI A, 01h   ;vloz 01 do A
       MVI B, 02h   ;vloz 02 do B
       ADD B        ;A = A + B
```

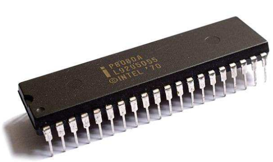
Assembler

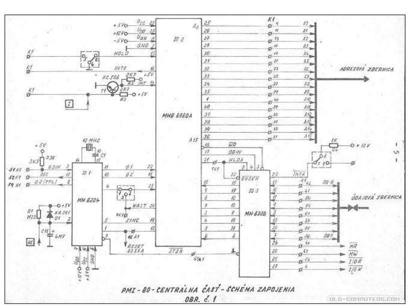
```
0000 3A 0D 00 47 3A 0E 00 80 0008
32 0F 00 76
```

Mikroprocesor rozumie len strojovému kódu!

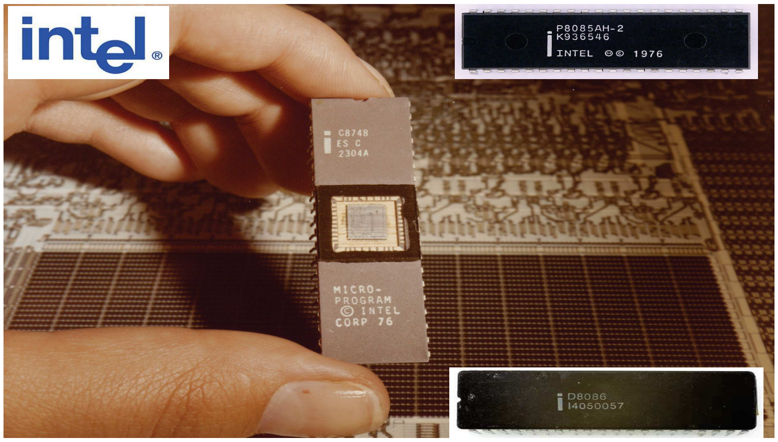


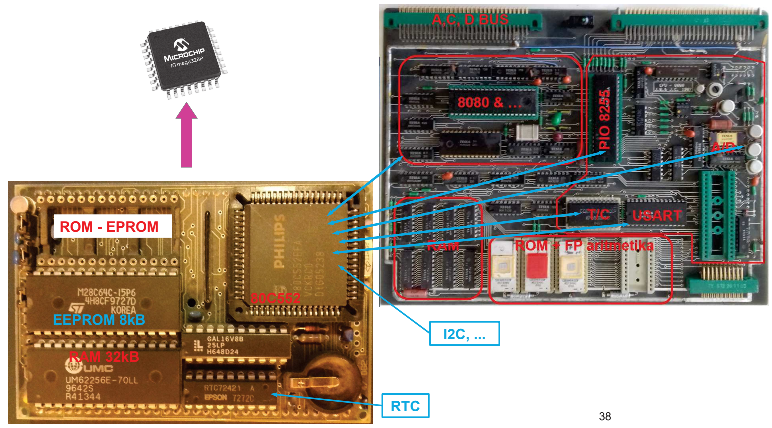




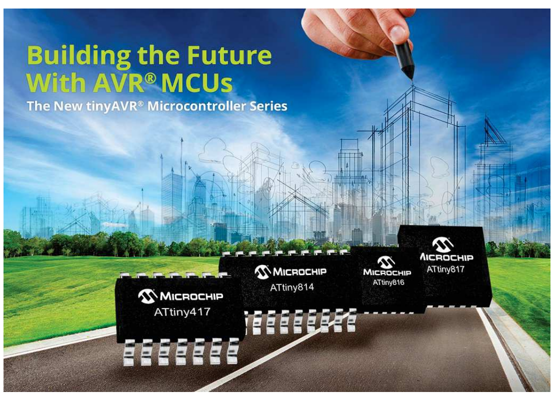


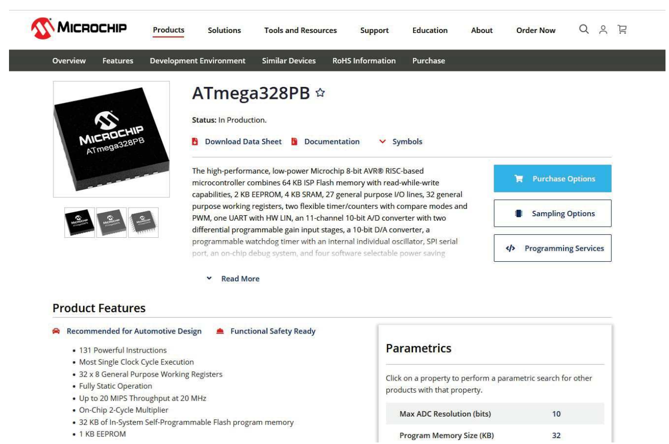
1976





38





AVR

1995 AVR architektúra: ***Alf-Egil Bogen*** a ***Vegard Wollan***, študenti na *Norwegian Institute of Technology v Trondheime*

Alf and **V**egard's **R**ISC processor
Advanced **V**irtual **R**ISC processor

original AVR MCU -- local ASIC house in Trondheim, Norway (now *Nordic Semiconductor*)

1996 technology was sold to Atmel from Nordic VLSI further developed by Bogen and Wollan at ***Atmel Norway***

jeden z prvých bol AT90S8515, 40-pin DIP pinout as an 8051 (polarity of the RESET line was opposite).

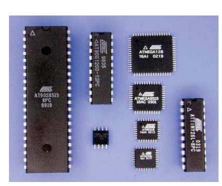
1997 začala výroba

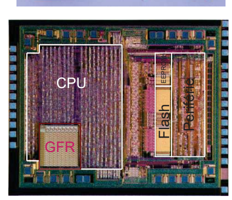
len do r. 2003, Atmel had shipped *500 million* AVR

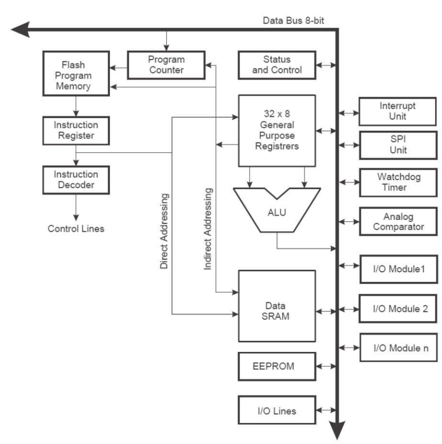
2019 Atmel -> ***Microchip Technology***

Procesor optimalizovaný pre jazyk C
+ RISC: väčšina inštrukcií trvá **1SC**
+ FLASH: rapídne zrýchlenie a zlacnenie vývoja

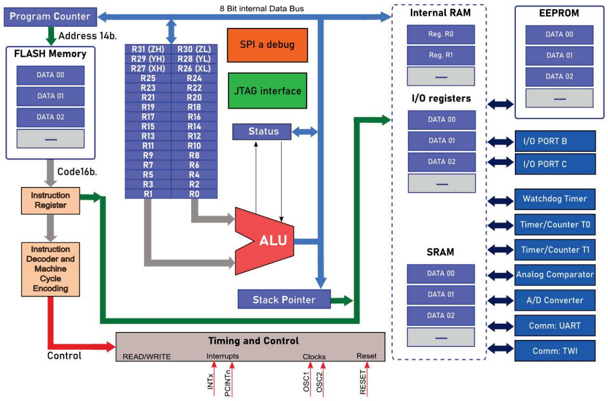
AVR - Architektúra







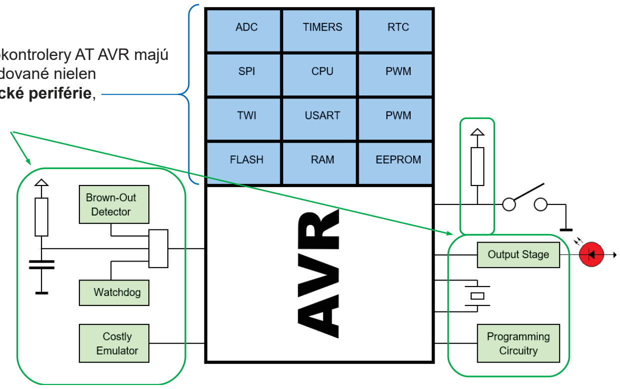
AVR -



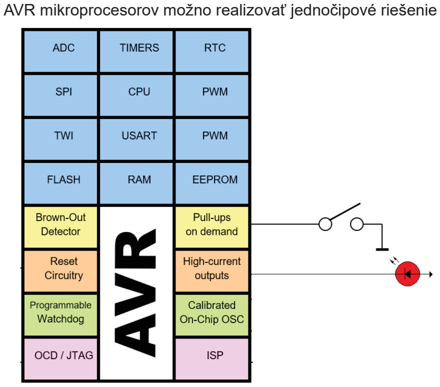
Pomocou AT AVR mikroprocesorov možno realizovať jednočipové riešenie

Mikrokontrolery AT AVR majú zabudované nielen **klasické periférie**,

ale aj

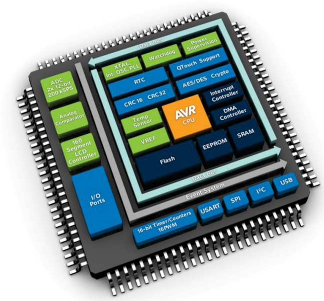


Pomocou AT AVR mikroprocesorov možno realizovať jednočipové riešenie

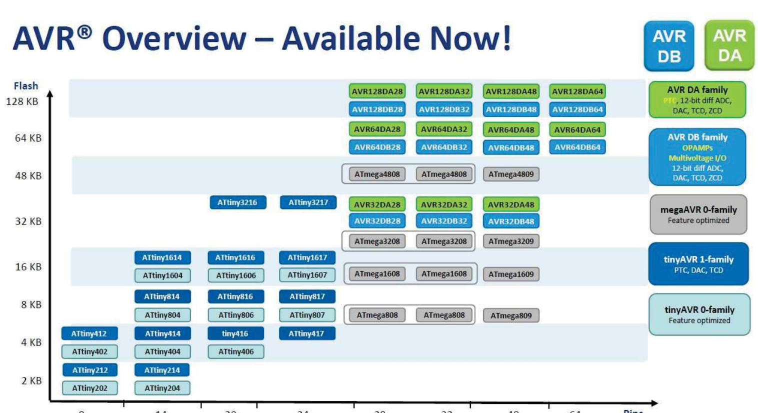


AVR families

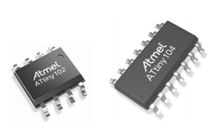
- **tinyAVR**
- **megaAVR**
- **XMEGA** (real time, USB, comm)
- **AVR UC3 L3** (battery, low power)
- **avr32**

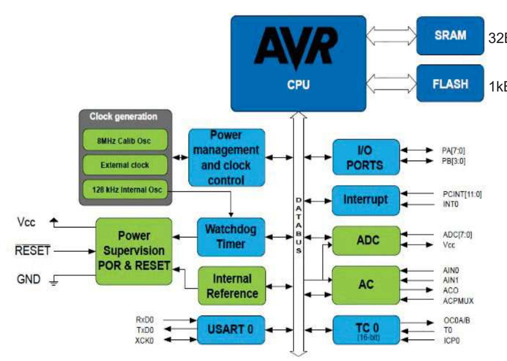


AVR® Overview – Available Now!



ATtiny

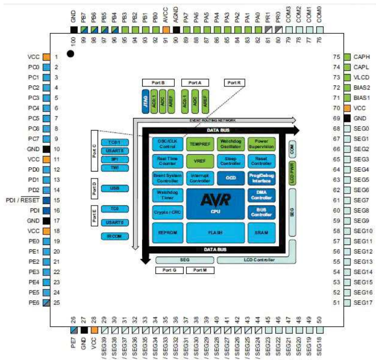




Microchip Technology ATtiny102/104 AVR 8-Bit Microcontrollers

XMEGA

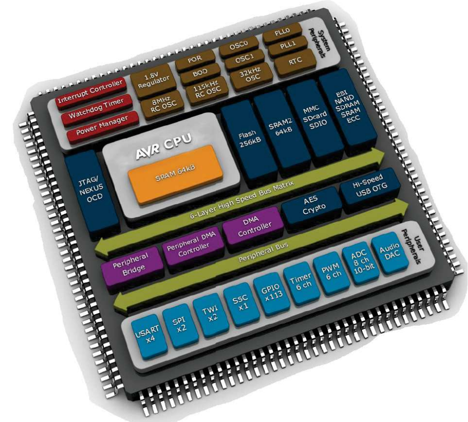
128 kB FLASH
8 kB boot
8 kB SRAM
2 kB EEPROM



Microchip Technology AVR® XMEGA® 8/16-bit Microcontroller 128B1

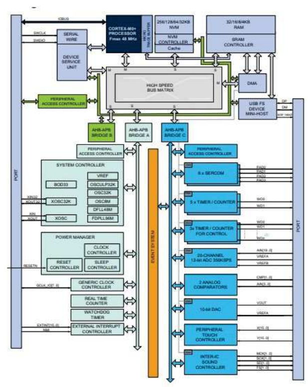
AVR32

256 kB FLASH
64 kB SRAM
SDcard



Microchip AVR® 32-bit AVR UC3

SAM -> nie je AVR!

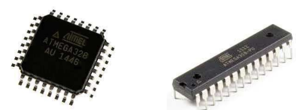


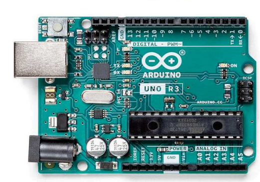
Microchip Technology ATSAMD21

ATmega328

32B

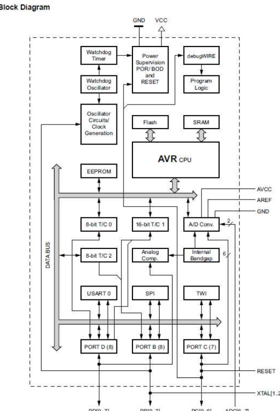
1kB



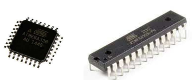


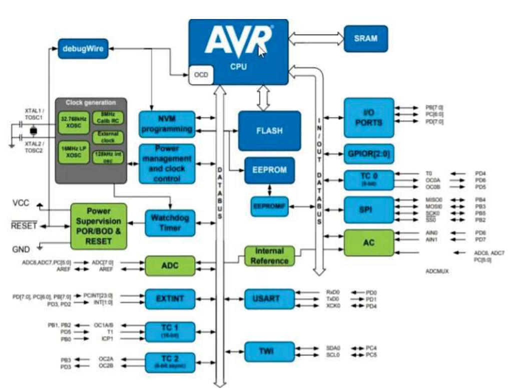
Microchip Technology ATmega328 AVR 8-Bit Microcontrollers

Figure 2-1. Block Diagram



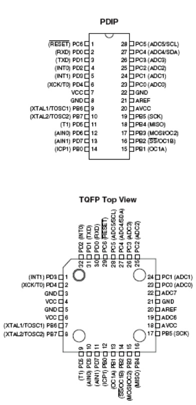
ATmega328

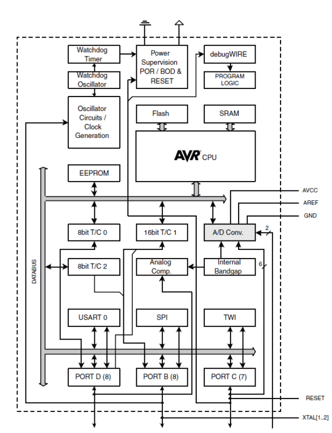




Microchip Technology ATmega328 AVR 8-Bit Microcontrollers

ATMEGA8





ATMEGA128

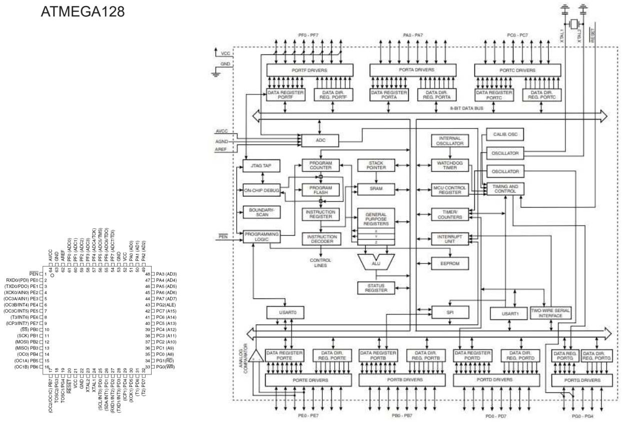
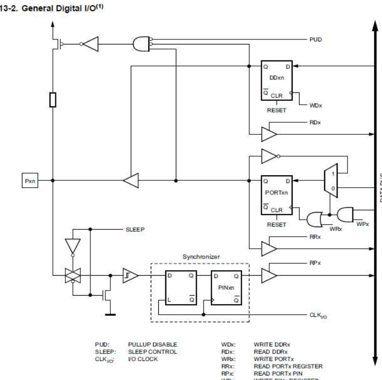
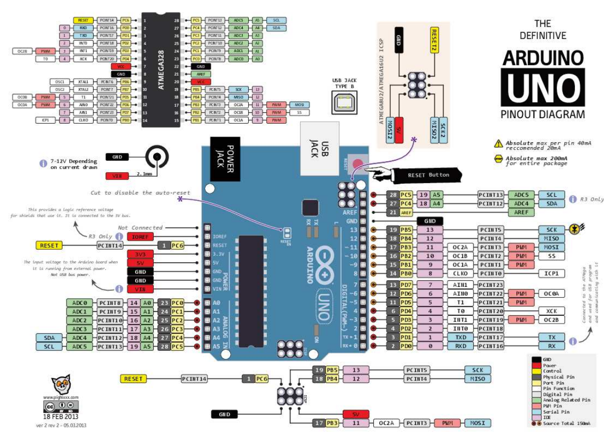


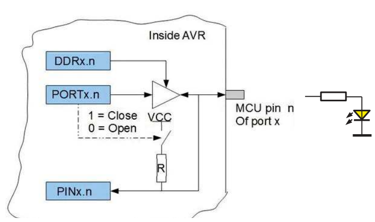
Figure 13-2. General Digital I/O[1]





Vstupy a výstupy (I/O)

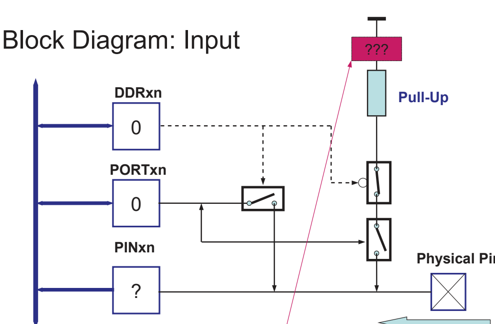
~~PORT A~~	PORT B.0
PORT B	PORT B.1
PORT C	PORT B.2
PORT D	PORT B.3
~~PORT E~~	PORT B.4
~~PORT F~~	PORT B.5
	~~PORT B.6~~
	~~PORT B.7~~



13.4.2 PORTB – The Port B Data Register

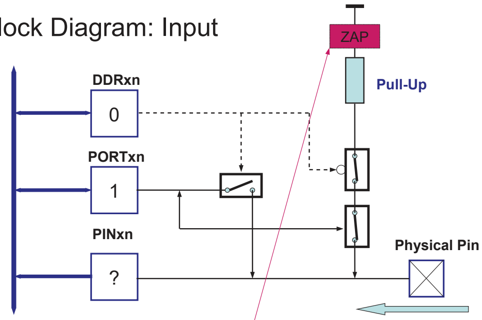
Bit	7	6	5	4	3	2	1	0	
0x05 (0x25)	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	PORTB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

I/O Pin Block Diagram: Input



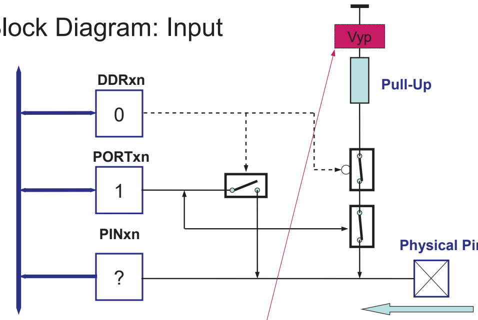
DDRxn	I/O	PORTxn	PUD (MCUCR)	Pull-up	Poznámka
0	In	0	X	No	Tri state (Hi-Z)

I/O Pin Block Diagram: Input



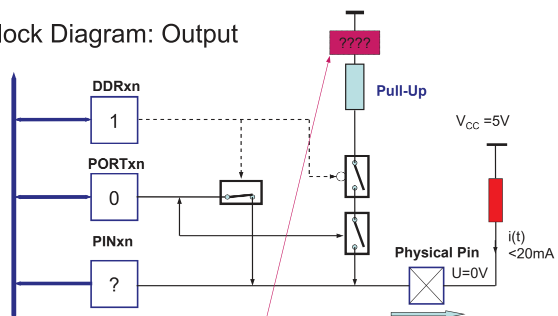
DDRxn	I/O	PORTxn	PUD (MCUCR)	Pull-up	Poznámka
0	In	1	0	Yes	Zdroj prudu (malého), ak „pulled down“

I/O Pin Block Diagram: Input



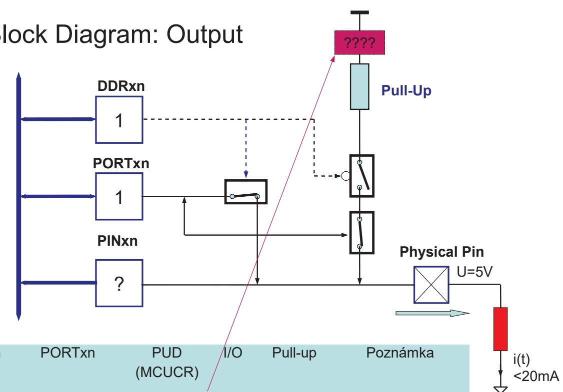
DDRxn	I/O	PORTxn	PUD (MCUCR)	Pull-up	Poznámka
0	In	1	1	No	Tri state (Hi-Z)

I/O Pin Block Diagram: Output



DDRxn	I/O	PORTxn	PUD (MCUCR)	Pull-up	Poznámka
1	Out	0	X	No	Out - L (Sink)

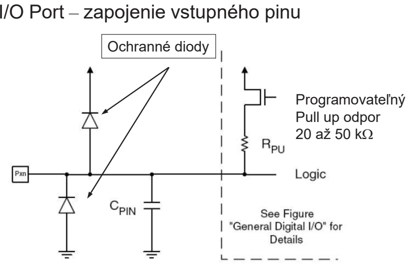
I/O Pin Block Diagram: Output



DDRxn	PORTxn	PUD (MCUCR)	I/O	Pull-up	Poznámka
1=Out	1=Pull-Up (off)	X	Out	No	Out - H (Source)

I/O Port – zapojenie vstupného pinu

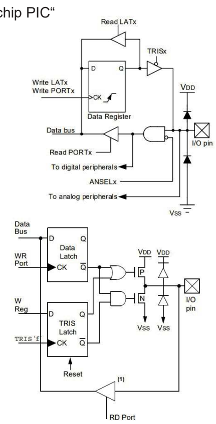
„ATMEL AVR“



?? $U_{pin} > V_{cc}$??

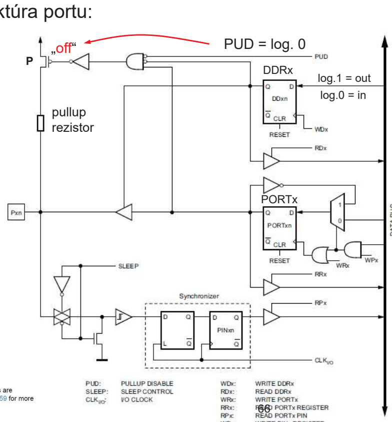
?? $U_{pin} < V_{ss}$??

„Microchip PIC“



I/O Port – novšia architektúra portu:

DDRx RD/RW
PORTx RD/WR
PINx RD/(WR)
x = A,B,.. F



MCUCR – MCU Control Register

Bit	7	6	5	4	3	2	1	0	
0x35 (0x55)	–	BODS	BODSE	PUD	–	–	IVSEL	IVCE	MCUCR
Read/Write	R	R	R	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

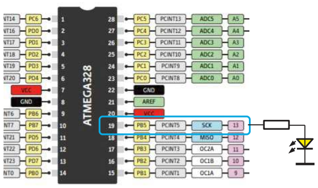
- **Bit 4 – PUD: Pull-up Disable**

When this bit is written to one, the pull-ups in the I/O ports are disabled even if the DDxn and PORTxn registers are configured to enable the pull-ups ({DDxn, PORTxn} = 0b01). See Section 13.2.1 “Configuring the Pin” on page 59 for more details about this feature.

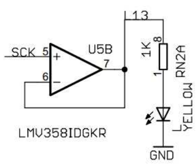
Programovanie I/O bitov v Arduino C

```
// initialize digital pin LED_BUILTIN as an output.
pinMode(LED_BUILTIN, OUTPUT);
          =13
// turn the LED on (HIGH is the voltage level)
digitalWrite(LED_BUILTIN, HIGH);

// turn the LED off by making the voltage LOW
digitalWrite(LED_BUILTIN, LOW);
```

St. ver.



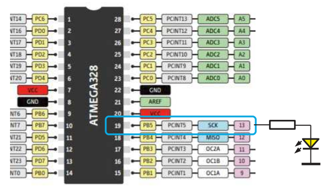
Nov. ver.



Programovanie I/O bitov v assembleri

```
START:      SBI 0x04,5     ; DDRB.5 = 1 (t.j. Output)
LOOP:       SBI 0x05,5     ; PORTB.5 = 1 (t.j. High, rozsviet LED)
            CBI 0x05,5     ; PORTB.5 = 0 (t.j. Low, zhasni LED)
```

St. ver.



13.4.2 PORTB – The Port B Data Register

Bit	7	6	5	4	3	2	1	0	
0x05 (0x25)	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	PORTB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

13.4.3 DDRB – The Port B Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x04 (0x24)	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	DDRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

13.4.4 PINB – The Port B Input Pins Address

Bit	7	6	5	4	3	2	1	0	
0x03 (0x23)	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	PINB
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

Programovanie I/O bitov v jazyku C

```
// initialize digital pin LED_BUILTIN as an output.
pinMode(LED_BUILTIN, OUTPUT);
```

DDRB |= (1 << DDRB5);

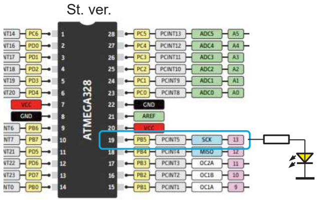
```
// turn the LED on (HIGH is the voltage level)
digitalWrite(LED_BUILTIN, HIGH);
```

PORTB |= (1 << PORTB5);

```
// turn the LED off by making the voltage LOW
digitalWrite(LED_BUILTIN, LOW);
```

PORTB &= ~(1 << PORTB5);

St. ver.



Programovanie I/O bitov v jazyku C

```
DDRB |= (1 << DDRB5);

DDRB  = DDRB | (1 << DDRB5);
```

	7	6	5	4	3	2	1	0
DDRB	X	X	?	X	X	X	0	1
OR	0	0	1	0	0	0	0	0
=	X	X	1	X	X	X	0	1

0	0	0	0	0	0	0	1

1 << 5

0	0	1	0	0	0	0	0

```
PORTB &= ~(1 << PORTB5);

PORTB  = PORTB  &  ~(1 << PB5);
```

	7	6	5	4	3	2	1	0
PORTB	X	X	?	X	X	X	0	1
&	1	1	0	1	1	1	1	1
=	X	X	0	X	X	X	0	1

1 << 5

0	0	1	0	0	0	0	0

~ (1 << 5)

1	1	0	1	1	1	1	1

https://senzor.robotika.sk/sensorwiki/index.php/AVR_Bit_Magic