Prednáška 1.

Distribuované vnorené počítačové systémy

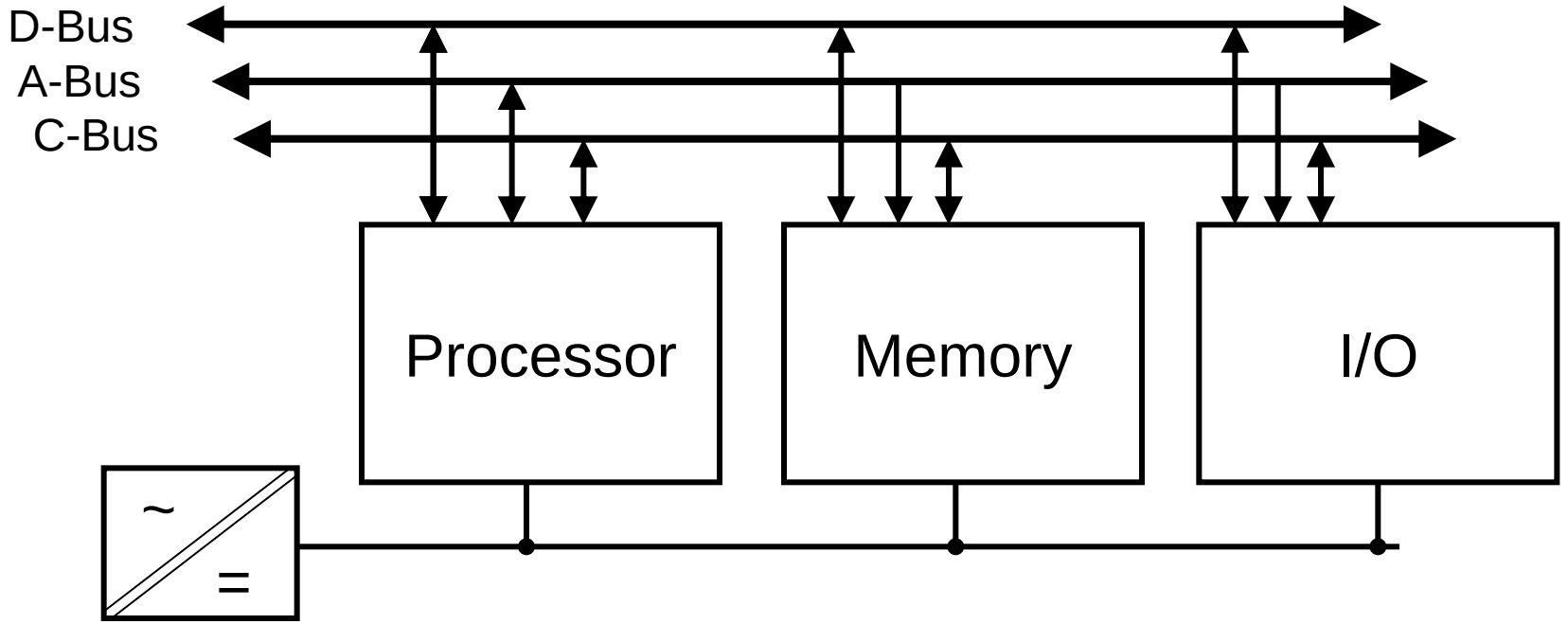
Distributed Embedded Computer System

Architektúra mikroprocesorov

http://ap.urpi.elf.stuba.sk/

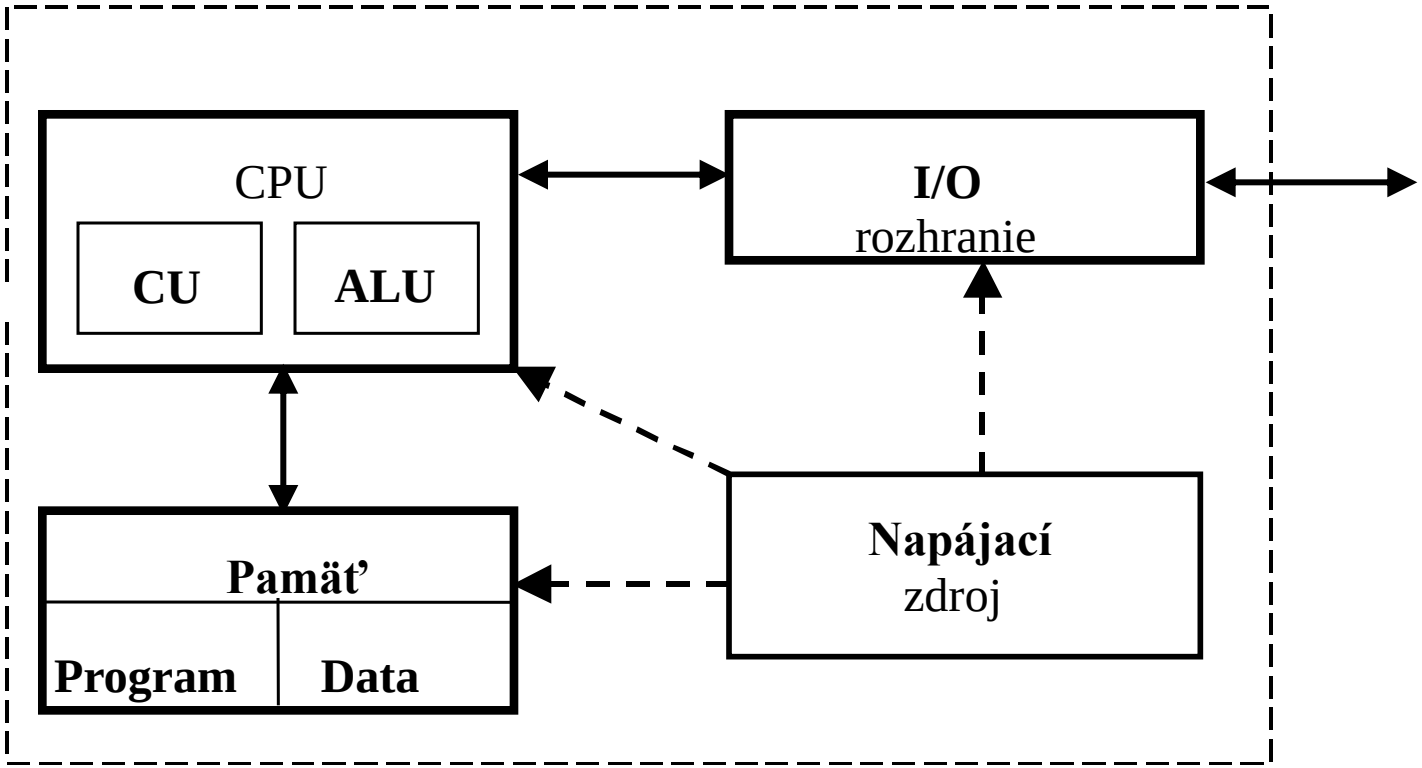
Štruktúra počítača

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



Von Neumann*ov* počítač

Princetonská architektúra



Von Neumann*ov* **počítač** – „vlastnosti“

Požadujeme: *rýchly procesor a veľkú pamäť =>*

Presúvanie informácií: *„von Neumannov bottleneck“ – John Backus 1977*

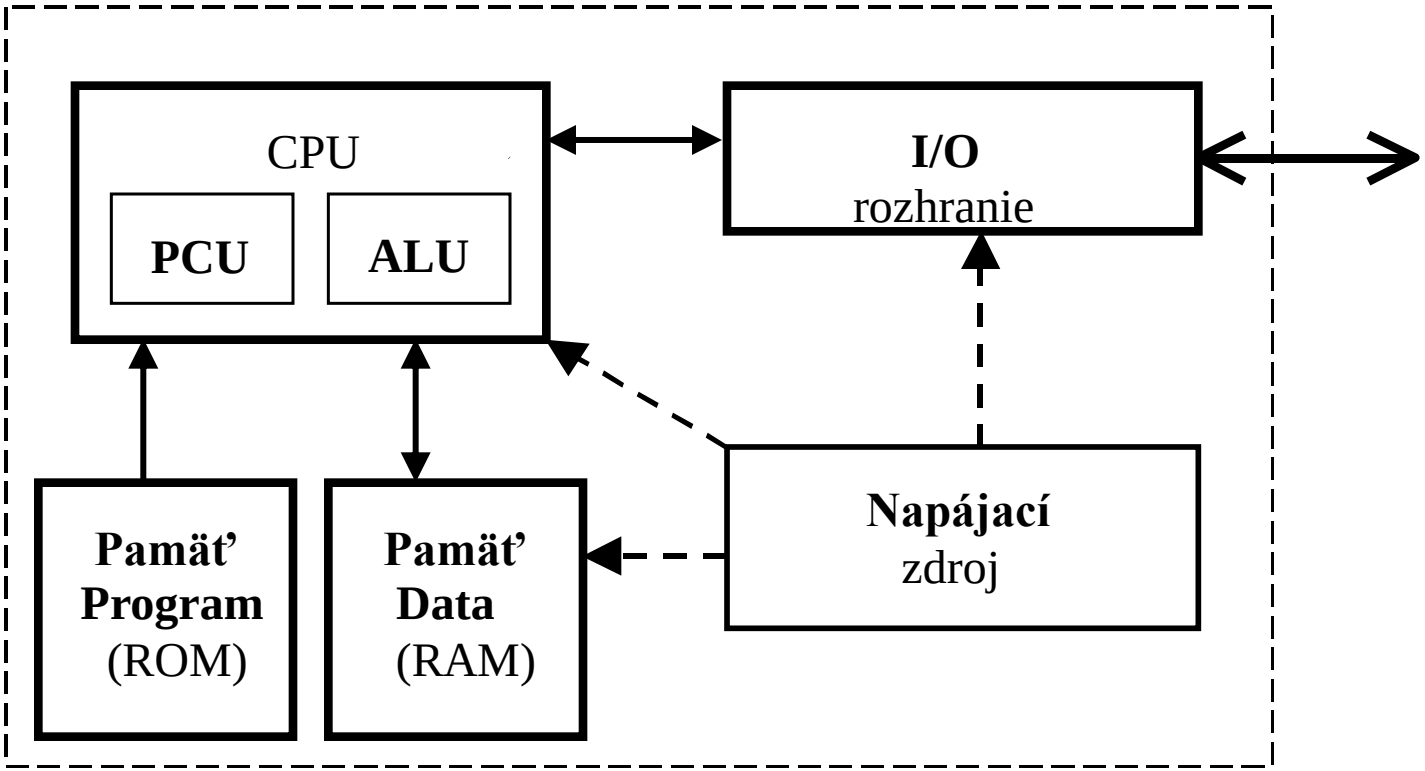
Vykonávanie programu (sériové):

- *Výber inštrukcie*
- *Dekódovanie inštr. a výber operandov*
- *Vykonanie operácie*

Program a Dáta: *sú uložené v pamäti*

Inštrukcie, Dáta (čísla, znaky, ...), adresy: *sú ukladané ako dvojkové čísla.*

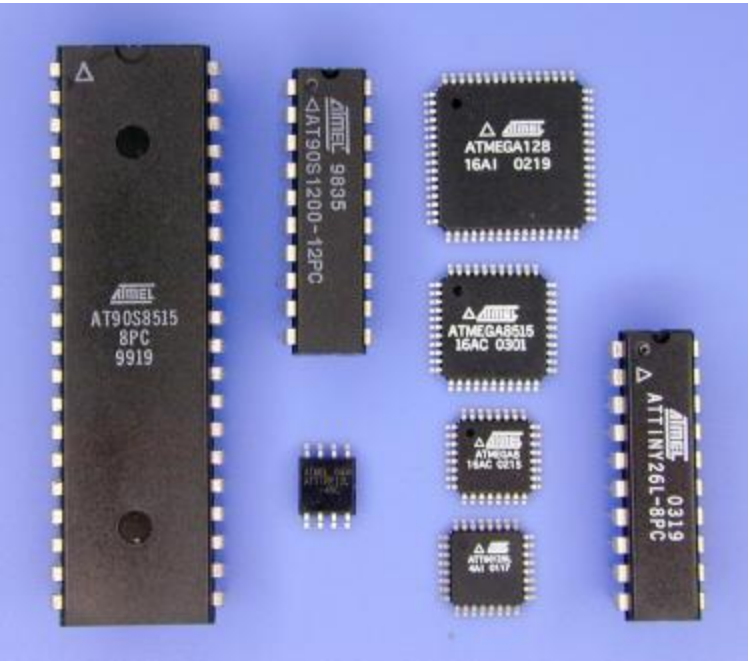
Hardvardská architektúra (Aiken)

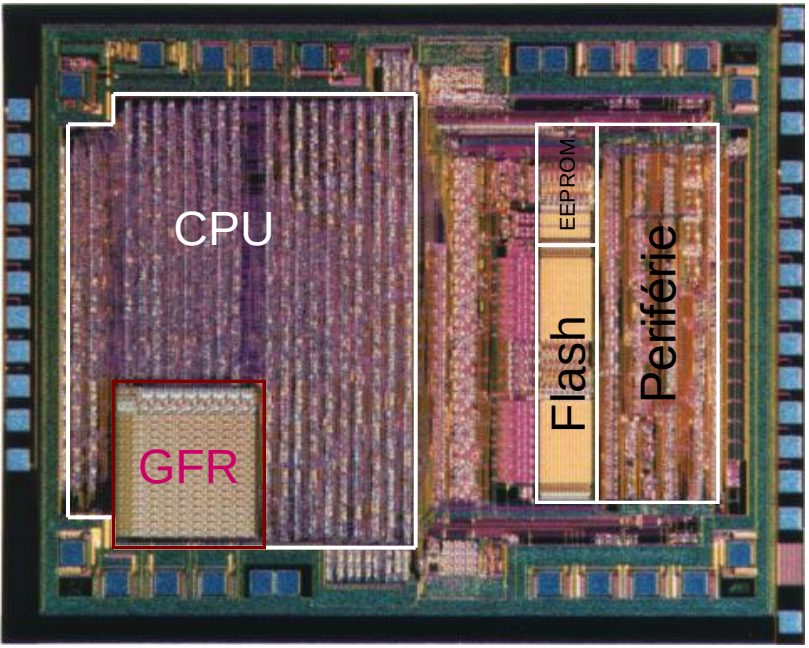


Mikrokontroler

- Počítač je vytvorený z: CPU, Pamätí, Periférií, I/O rozhrania
- Mikrokontroler je vlastne malý počítač na jednom čipe
- Programuje sa podobne ako počítač. Spracováva vstupné dáta, realizuje výpočty a generuje výstupy
- Mikrokontrolery sú súčasťou tzv. „Embedded systems“

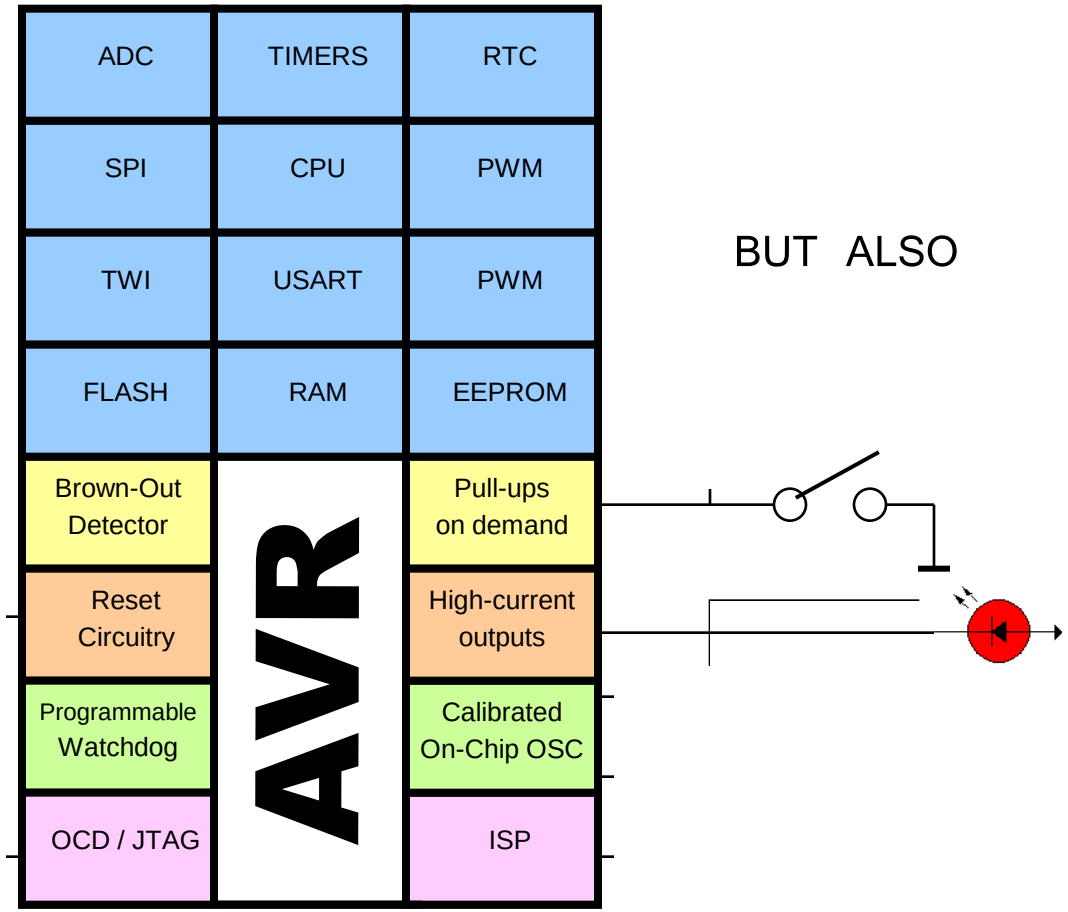
AVR - Architektúra





AVR, for a **true** single-chip solution

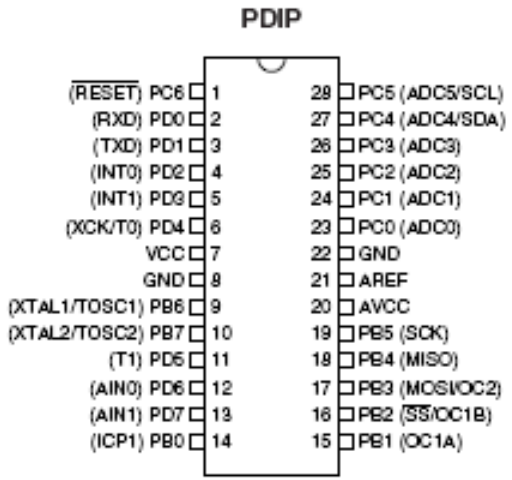
AVR does not only give you that

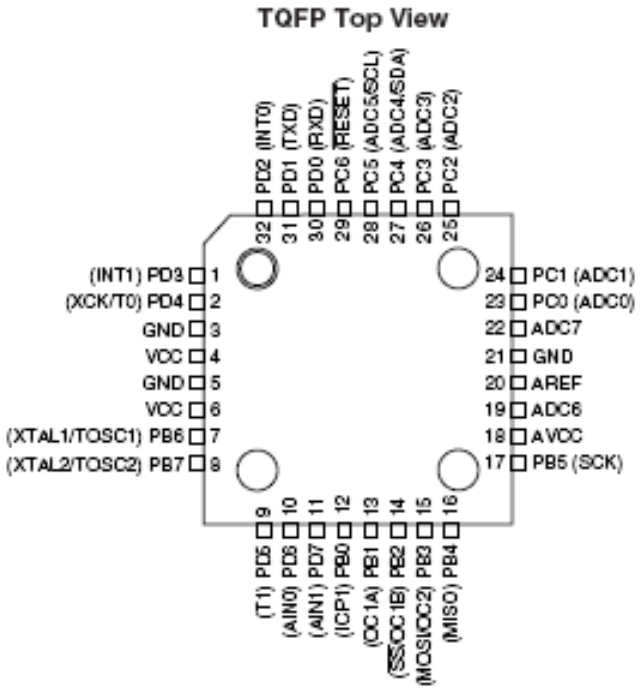


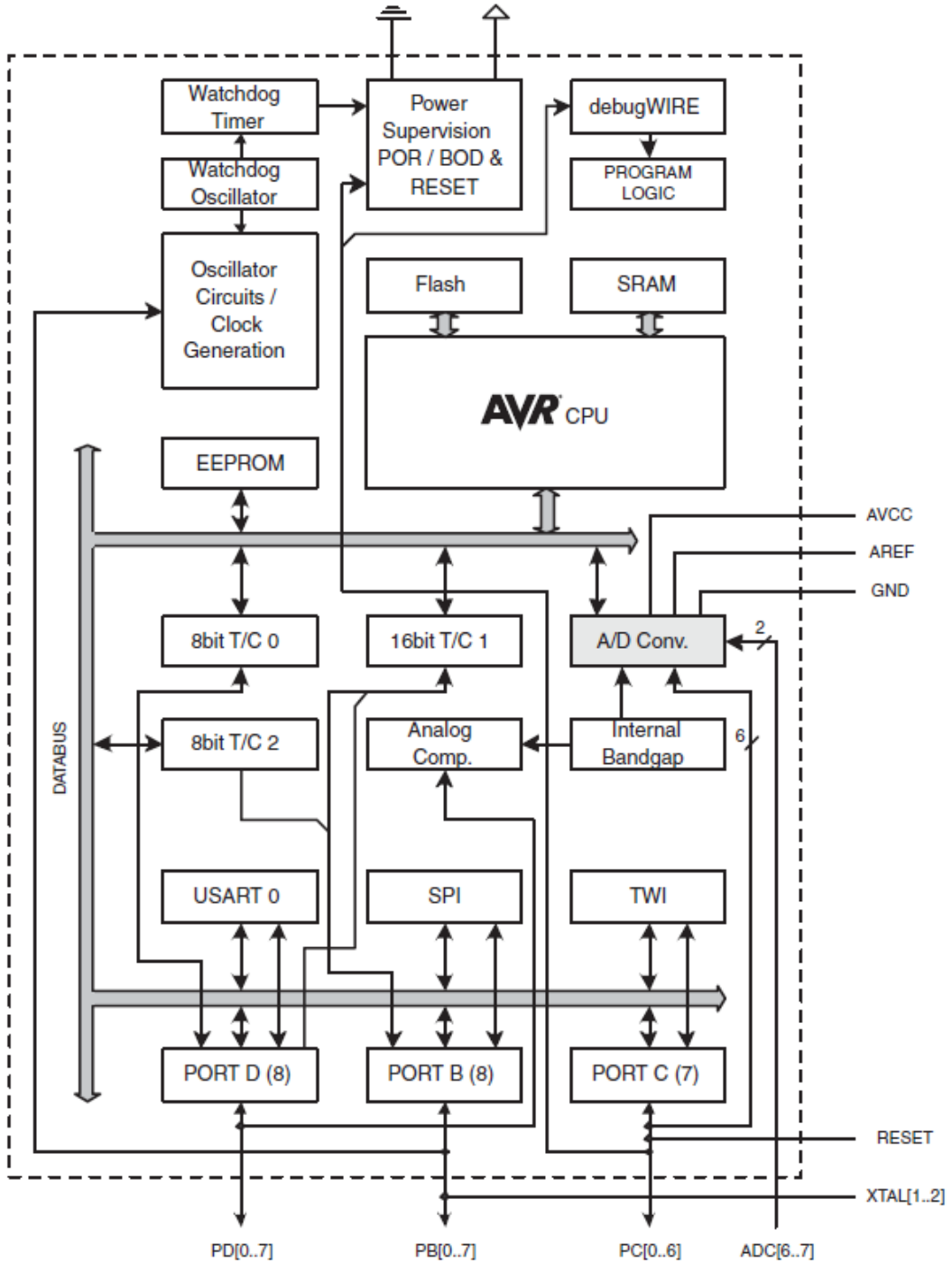
BUT ALSO

TO open the way to a **true** single-chip solution !

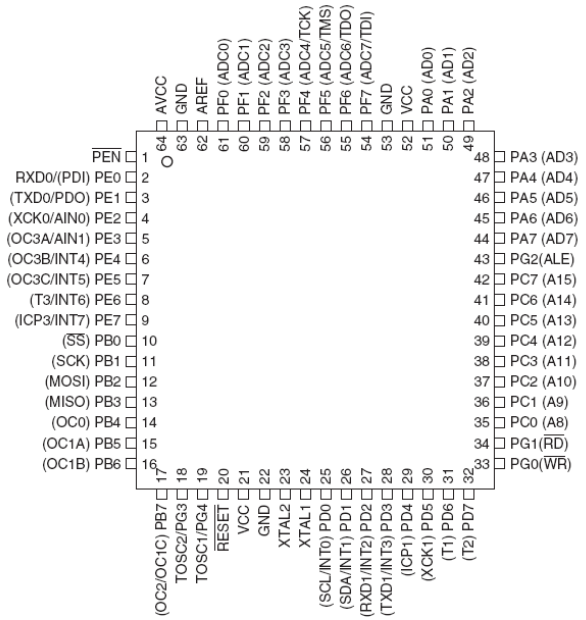
ATMEGA8

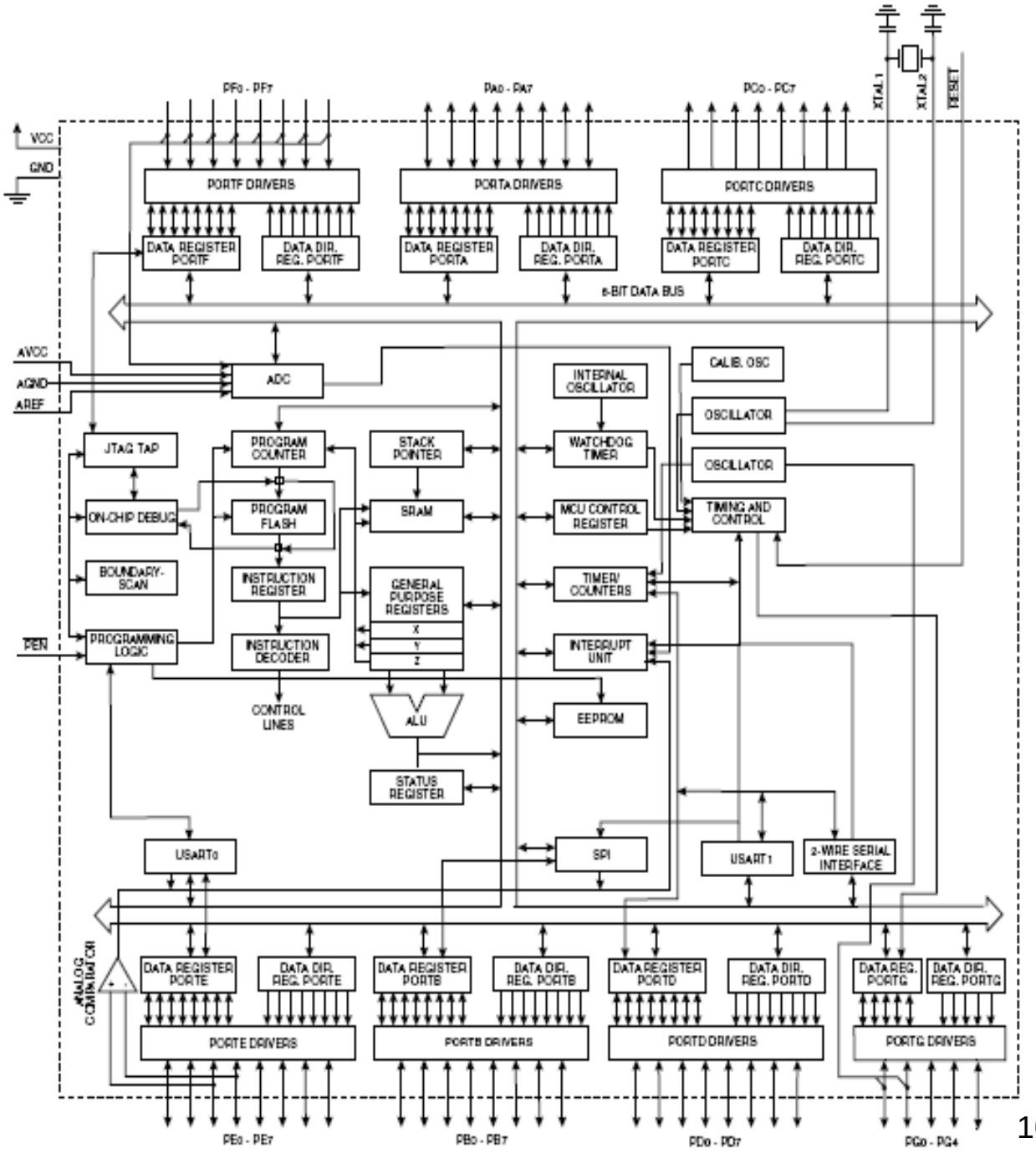

PDIP
(RESET) PC6 1
(RXD) PD0 2
(TXD) PD1 3
(INT0) PD2 4
(INT1) PD3 5
(XCK/T0) PD4 6
VCC 7
GND 8
(XTAL1/TOSC1) PB6 9
(XTAL2/TOSC2) PB7 10
(T1) PD5 11
(AIN0) PD6 12
(AIN1) PD7 13
(ICP1) PB0 14
28 PC5 (ADC5/SCL)
27 PC4 (ADC4/SDA)
26 PC3 (ADC3)
25 PC2 (ADC2)
24 PC1 (ADC1)
23 PC0 (ADC0)
22 GND
21 AREF
20 AVCC
19 PB5 (SCK)
18 PB4 (MISO)
17 PB3 (MOSI/OC2)
16 PB2 (SS/OC1B)
15 PB1 (OC1A)


TQFP Top View
32 PD2 (INT0)
31 PD1 (TXD)
30 PD0 (RXD)
29 PC6 (RESET)
28 PC5 (ADC5/SCL)
27 PC4 (ADC4/SDA)
26 PC3 (ADC3)
25 PC2 (ADC2)
(INT1) PD3 1
(XCK/T0) PD4 2
GND 3
VCC 4
GND 5
VCC 6
(XTAL1/TOSC1) PB6 7
(XTAL2/TOSC2) PB7 8
24 PC1 (ADC1)
23 PC0 (ADC0)
22 ADC7
21 GND
20 AREF
19 ADC6
18 AVCC
17 PB5 (SCK)
9 (T1) PD5
10 (AIN0) PD6
11 (AIN1) PD7
12 (ICP1) PB0
13 (OC1A) PB1
14 (SS/OC1B) PB2
15 (MOSI/OC2) PB3
16 (MISO) PB4


Watchdog Timer
Watchdog Oscillator
Power Supervision POR / BOD & RESET
debugWIRE
PROGRAM LOGIC
Oscillator Circuits / Clock Generation
Flash
SRAM
AVR CPU
EEPROM
8bit T/C 0
16bit T/C 1
A/D Conv.
8bit T/C 2
Analog Comp.
Internal Bandgap
USART 0
SPI
TWI
PORT D (8)
PORT B (8)
PORT C (7)
DATABUS
AVCC
AREF
GND
RESET
XTAL[1..2]
PD[0..7]
PB[0..7]
PC[0..6]
ADC[6..7]

ATMEGA128


AVCC
GND
AREF
PF0 (ADC0)
PF1 (ADC1)
PF2 (ADC2)
PF3 (ADC3)
PF4 (ADC4/TCK)
PF5 (ADC5/TMS)
PF6 (ADC6/TDO)
PF7 (ADC7/TDI)
GND
VCC
PA0 (AD0)
PA1 (AD1)
PA2 (AD2)
PEN
RXD0/(PDI) PE0
(TXD0/PDO) PE1
(XCK0/AIN0) PE2
(OC3A/AIN1) PE3
(OC3B/INT4) PE4
(OC3C/INT5) PE5
(T3/INT6) PE6
(ICP3/INT7) PE7
(SS) PB0
(SCK) PB1
(MOSI) PB2
(MISO) PB3
(OC0) PB4
(OC1A) PB5
(OC1B) PB6
PA3 (AD3)
PA4 (AD4)
PA5 (AD5)
PA6 (AD6)
PA7 (AD7)
PG2(ALE)
PC7 (A15)
PC6 (A14)
PC5 (A13)
PC4 (A12)
PC3 (A11)
PC2 (A10)
PC1 (A9)
PC0 (A8)
PG1(RD)
PG0(WR)
(OC2/OC1C) PB7
TOSC2/PG3
TOSC1/PG4
RESET
VCC
GND
XTAL2
XTAL1
(SCL/INT0) PD0
(SDA/INT1) PD1
(RXD1/INT2) PD2
(TXD1/INT3) PD3
(ICP1) PD4
(XCK1) PD5
(T1) PD6
(T2) PD7

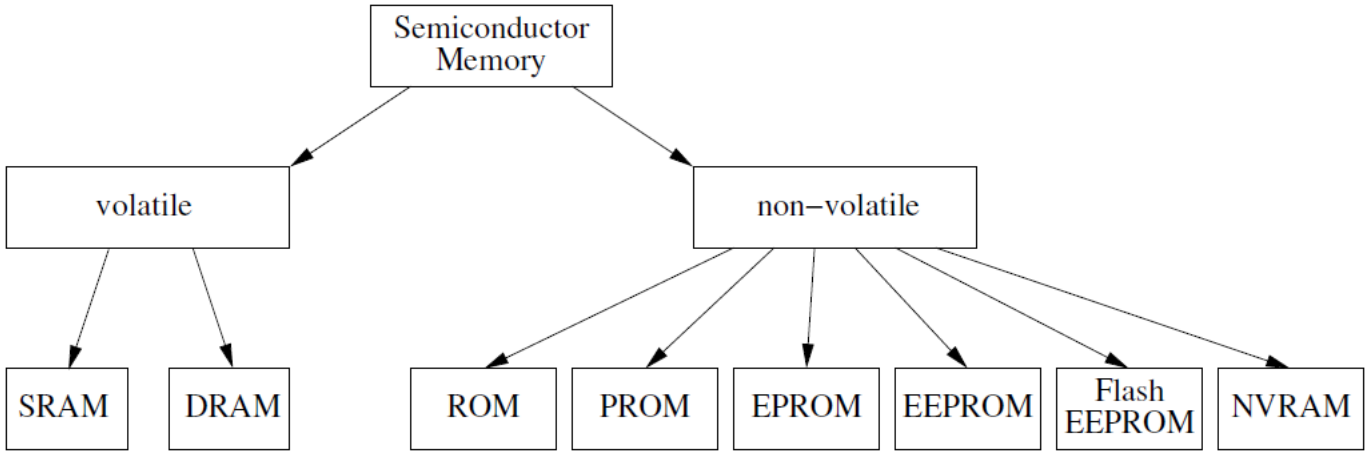

PF0 - PF7
PA0 - PA7
PC0 - PC7
XTAL1
XTAL2
RESET
VCC
GND
PORTF DRIVERS
PORTA DRIVERS
PORTC DRIVERS
DATA REGISTER PORTF
DATA DIR. REG. PORTF
DATA REGISTER PORTA
DATA DIR. REG. PORTA
DATA REGISTER PORTC
DATA DIR. REG. PORTC
8-BIT DATA BUS
AVCC
AGND
AREF
ADC
INTERNAL OSCILLATOR
CALIB. OSC
OSCILLATOR
OSCILLATOR
JTAG TAP
PROGRAM COUNTER
STACK POINTER
WATCHDOG TIMER
ON-CHIP DEBUG
PROGRAM FLASH
SRAM
MCU CONTROL REGISTER
TIMING AND CONTROL
BOUNDARY-SCAN
INSTRUCTION REGISTER
GENERAL PURPOSE REGISTERS
X
Y
Z
TIMER/ COUNTERS
PEN
PROGRAMMING LOGIC
INSTRUCTION DECODER
INTERRUPT UNIT
CONTROL LINES
ALU
EEPROM
STATUS REGISTER
USART0
SPI
USART1
2-WIRE SERIAL INTERFACE
ANALOG COMPARATOR
DATA REGISTER PORTE
DATA DIR. REG. PORTE
DATA REGISTER PORTB
DATA DIR. REG. PORTB
DATA REGISTER PORTD
DATA DIR. REG. PORTD
DATA REG. PORTG
DATA DIR. REG. PORTG
PORTE DRIVERS
PORTB DRIVERS
PORTD DRIVERS
PORTG DRIVERS
PE0 - PE7
PB0 - PB7
PD0 - PD7
PG0 - PG4

- Harvardská 8-bitová RISC architektúra

- Oddelená pamäť programu a dát - samostatné zbernice pre program a dáta. Toto umožňuje súčasný prístup procesoru k obom pamätiam.
  - Jednu inštrukciu vykonávame a druhú môžeme vyberať. Navonok to vyzerá, ako keby bola v každom takte vykonaná jedna inštrukcia.
  - Iné mikropočítače pristupujú k týmto pamätiam po spoločnej zbernici. To znamená, že musia striedať vykonanie a výber inštrukcie.
- Veľký Accumulator
  - **32 * 8 bitov "General Purpose Registers" GPR**
  - GPR registre umožňujú vykonávanie jednoduchých operácií medzi sebou. To znamená redukujú sa požiadavky na prenos dát.
- Všetky inštrukcie sú 16 bitové(word),
- Jedno cyklový (SC) prístup k pamäti
- Jedno cyklová aritmetika
- ALU
- Stack Pointer
- Program Counter

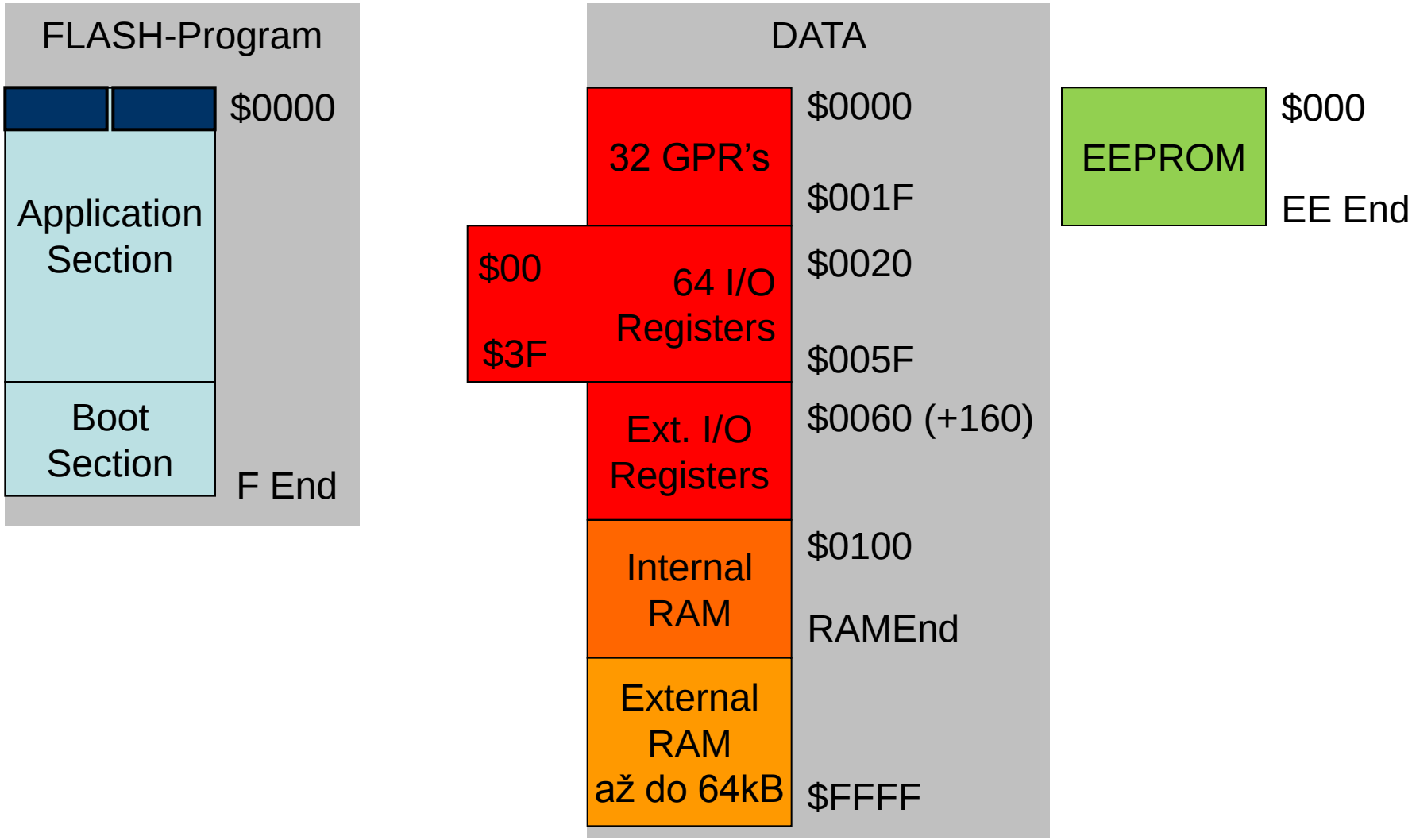
Memory – Pamäťový podsystém

- Register File (registre): Malá pamäť zabudovaná v CPU. „Slúži na krátkodobé odkladanie pre CPU
- Data Memory (RAM, pamäť dát): Väčšia pamäť na dlhšie ukladanie dát.
- Instruction Memory (ROM, FLASH, pamäť programu) : Veľká pamäť na uloženie inštrukcií



!!! Slovo *volatile* sa môže objaviť aj v HLL !!!

AVR mikrokontroler – Mapa pamäte

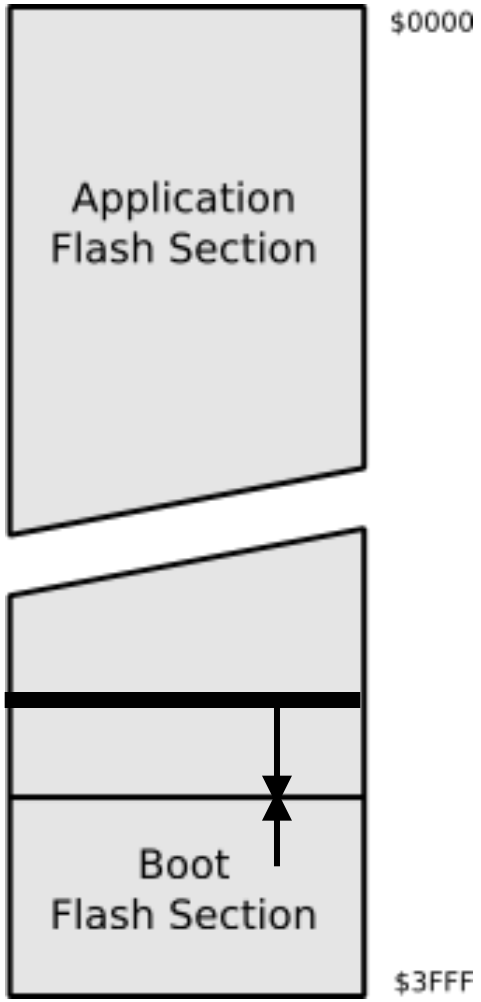


Programovateľná Flash pamäť (Non-volatile)

- Všetky AVR obvody majú preprogramovaťelnú **pamäť programu** FLASH.
- Napájanie 1,8V až 5,5V
- Stálosť údajov – cca 20rokov
- Počet Write/Erase cyklov – 10000
- Bezpečnostné poistky

Pamäť programu

- AVR program je uložený v nonvolatile (nezávislá od výpadku napájania) programovateľnej Flash pamäti. Rozdelená je do dvoch sekcií.
  - Prvá sekcia: Blok pamäte určený pre ukladanie aplikačných programov.
  - Druhá sekcia: Po pripojení napájania môže program uložený v tejto pamäti prepísať program v sekcii prvej.
- Program zapísaný do 2. sekcie sa zvykne nazývať **Bootloader**. Do aplikačnej časti pamäte môžeme zapísať program pomocou PROGRAMÁTORA alebo pomocou krátkeho programu zapísaného v 2. sekcii, ktorý prijíma dáta zo sériového portu. (*Popíš realizáciu a zabezpečenie*)



Ukladanie údajov do pamäte:

- Problém ukladania informácií, ak veľkosť údajového typu nie je zhodná s veľkosťou strojového slova.

Ukladanie a čítanie z pamäti v strojových slovách
(dané šírkou dátovej zbernice 16b = 2B, 32b = 4B, 64b = 8B)

- Poradie ukladania bytov údajového typu do pamäte.
- Pamäť je prístupná cez *adresu* – index na pole prvkov.
- Dáta možno vybrať (read) alebo uložiť (write) na konkrétnu adresu v pamäti.

$$\overset{3\ \ 2\ \ 1\ \ 0}{(305\ 419\ 896)_{10} = (12\ 34\ 56\ 78)_H}$$

Adresa	BE	LE	ME	
			2301	1032
0x1000	0x12 (MSB)	0x78	0x34	0x56
0x1001	0x34	0x56	0x12	0x78
0x1002	0x56	0x34	0x78	0x12
0x1003	0x78 (LSB)	0x12	0x56	0x34

Stredný endián – poradie bajtov 2301 (PDP11)
V architektúrach počítačov sa využívajú nasledujúce ukladacie endiány :
• len malý endián (Intel i386)
• len velký endián (MC 68030)
• Musí byť riešená konverzia rôznych endiánov, ak niektorý podsystém počítača pracuje v inom ukladacom režime.

• AVR procesory ukladajú údaje do FLASH (?BE/LE ?)

Dátum:

US : middle - endian

Month, Day, Year (May, 24th,2006 = 5/24/2006)

Europe : little - endian

Day, Month, Year (24th, May, 2006 = 24/ 5/2006)

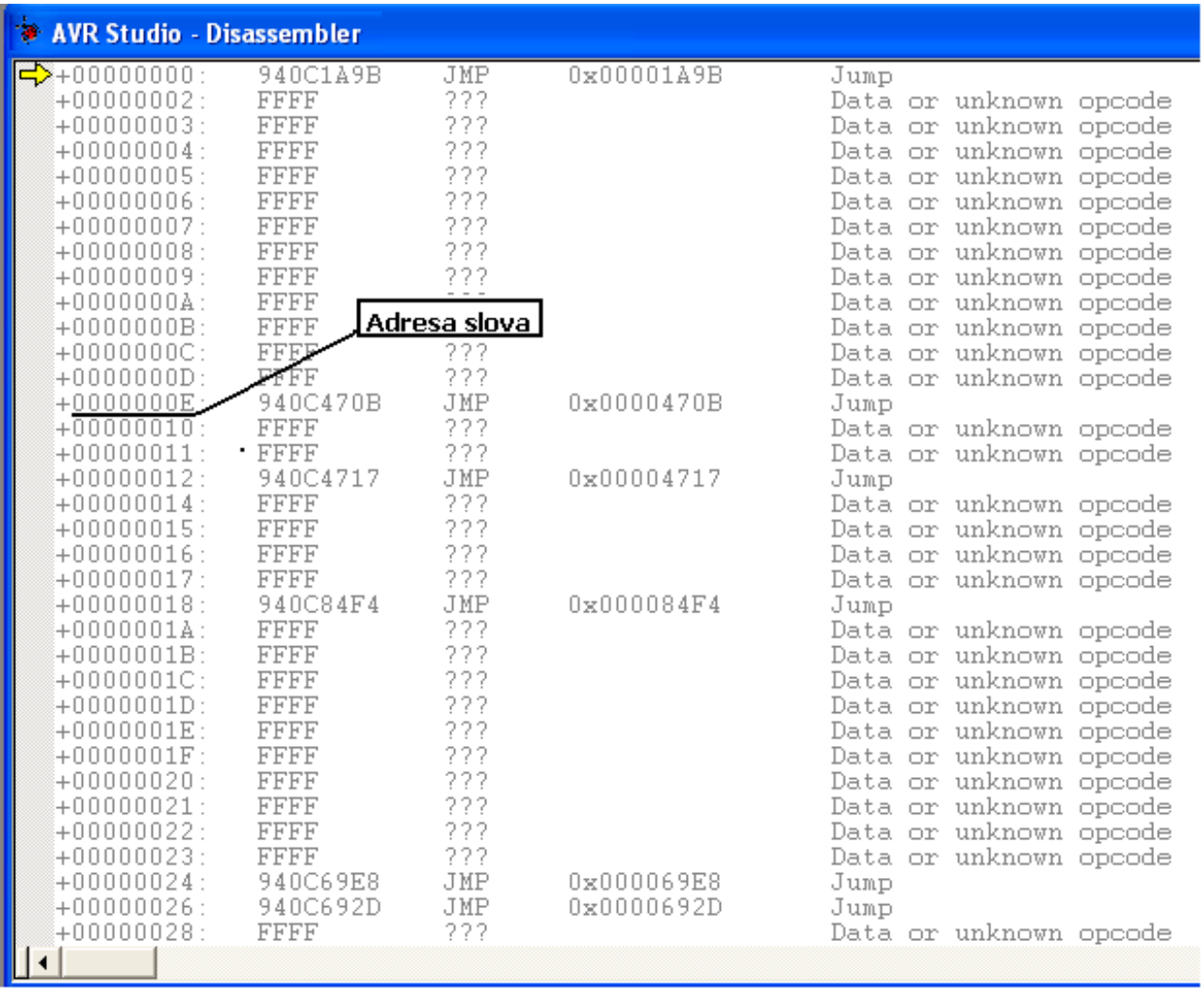
China, Japan & ISO 8601: big - endian

Year, Month, Day (2006, May, 24th = 2006-05-24)

Poznámky k endianom:

Neda sa povedať, že niektorý endian je výhodnejší voči inému, len ak zapíšeme dátum v big endian – ľahšie sa triedia položky.

Ak prenášame súbory medzi počítačmi s rôznymi endianmi, treba robiť vykonať transformáciu.



Výpis HEX-a súboru ATMEGA 128 (začiatok)

:04 0000 00 0C94 9B1A A7
:04 001C 00 0C94 0B47 EE
:04 0024 00 0C94 1747 DA
:04 0030 00 0C94 F484 B4
:08 0048 00 0C94 E869 0C94 2D69 89
:04 0054 00 0C94 2B48 95

Adresa Byte-u

JMP = 940C = 1001 0100 0000 1100

$2^{(6+16)} = 4*2^{(20)} = 4MB$

Pamäť dát

Dátová pamäť je realizovaná ako volatile RAM.
Organizovaná je po Bytoch – 8bitov.

Byte = 8 bitov.

Do bytu môžeme uložiť 256 rozdielnych hodnôt

Do bitov zapisujeme log.1, resp. log.0 – binárne.

Obsah Bytu môžeme zapísať ako dva HEXa znaky.

Pr: 11110011 - binárne, 243 - decimálne, F3 - hexadecimalne. Kvôli rozlíšeniu zapisujeme 0xF3 (základ čísla je 16).

Adresa pamäťového miesta sa väčšinou píše hexadecimálne. Aby sme odlíšili adresu od dát zapisujeme ju s $ na začiatku: napr. $02AF.

Adresa	MSB							LSB	Hex	Dec
	7	6	5	4	3	2	1	0		
$0000	1	0	1	0	0	1	0	1	A5	165
$0001	0	1	1	1	1	1	1	1	7F	127
$0002	1	0	0	0	0	0	0	1	81	129
$0003	1	1	0	0	1	1	0	1	CD	205

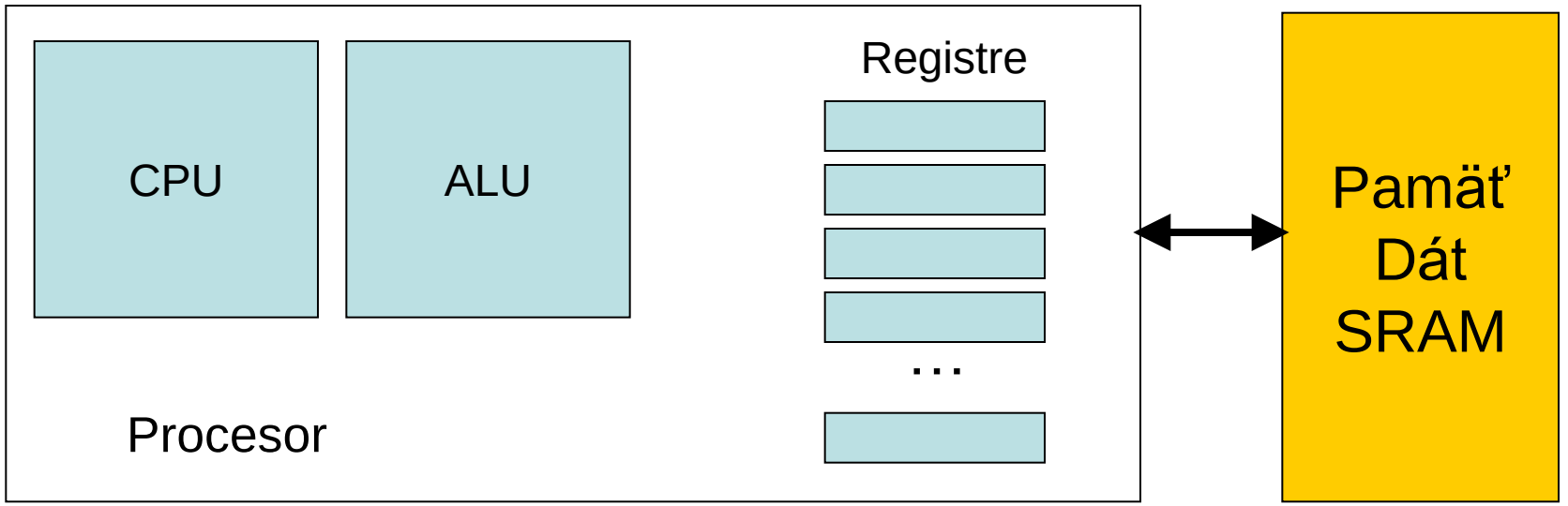
I/O Registre – porty (Read Modify Write (SBI,SCI))

- **Organizované** sú po **8b = 1B**
- **Označované sú písmenami A,B,C, ...**
- **Set/Clear I/O**
- Nastavovanie: log.1 a nulovanie: log.0 sa dá realizovať po bytoch, resp. po bitoch
- **DDRx Registre**
- Orientácia pinu, IN/OUT, sa nastavuje v reg. DDRx
- Ak je bit DDRx nastavený na log. 0 odpovedajúci pin portu je vstupný
- Ak je bit DDRx nastavený na log. 1 odpovedajúci pin portu je výstupný

- Ak DDRA = 0xF0 (1111 0000), to znamená bity 7-4 on PORTA sú výstupné a bity 3-0 asu vstupné.

- **Ďalšie funkcie portov ("funkcia má vyššiu prioritu")**
- Väčšina portov má okrem I/O funkcií aj ďalšie vlastnosti – funkcie.
- Periférií je viac ako pinov.
- U väčšiny procesorov je jeden port, jeho časť, vyhradená na vstup analógových signálov do A/D prevodníka
- Iné bity, iných portov, sú vo funkcii, napr.:
  - Dátovej riadiacej a adresnej zbernice,
  - vstupov a výstupov USART a I2C

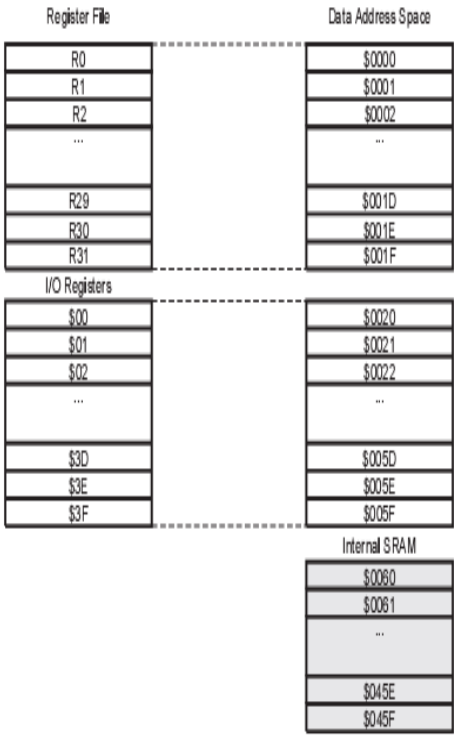
Procesor - Registre



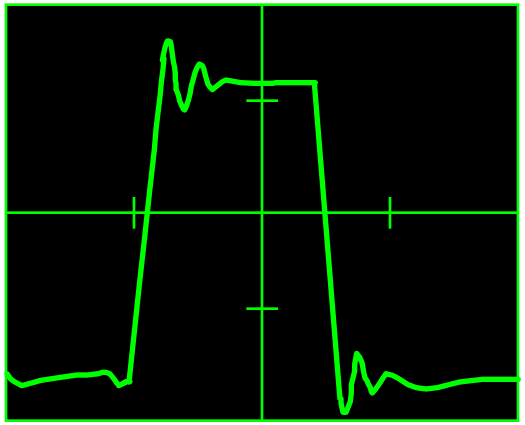
Do registra možno uložiť jeden 8b údaj.

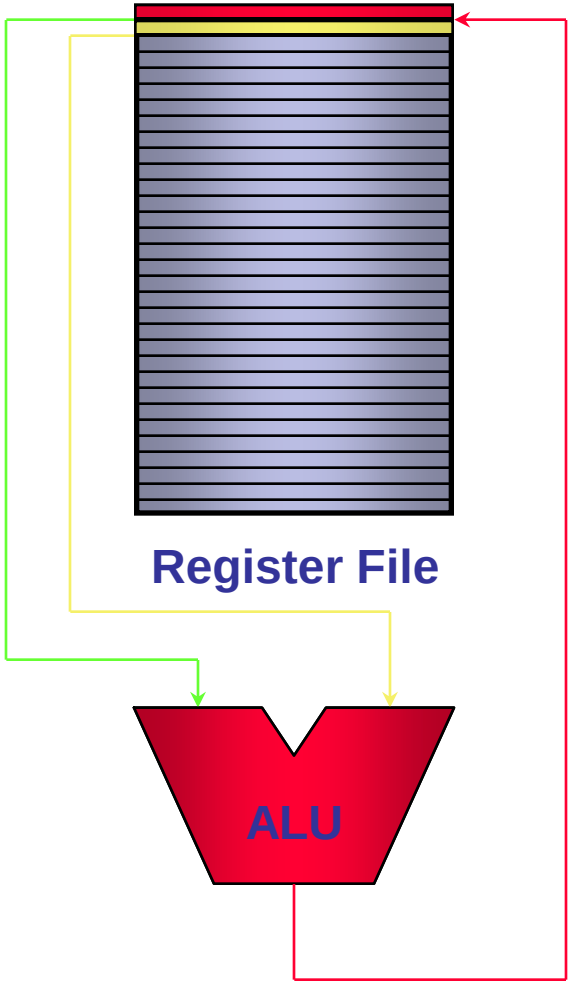
Prístup do registrov je rýchlejší
ako do pamäte dát.

Registre sú implementované na čipe.



Registre sú k ALU pripojené priamo





Registrové operácie trvajú jeden strojový cyklus

Load/Store architektúra

- podstatná výhoda RISC procesorov
- všetky aritmetické a logické operácie sa vykonávajú v GPR „general purpose registers“ (registre pre obecné použitie)
- AVR procesor môže adresovať až 64KB linearneho dátového
    - Laod/Store inštrukcií.
    - „direct“ alebo „indirect“ adresovanie
    - Pri nepriamom adresovaní, sa použije časť GPR ako smerníky na adresu.
    - ďalšie adresovacie módy: posunutie, post-increment or pre-decrement.

- GPR a I/O registre sú prístupné pomocou **mov/in/out** inštrukcií a sú mapované do DÁTOVÉHO pamäťového priestoru, to znamená, že sú prístupné aj pomocou Load/Store inštrukcií.

Mapovanie I/O obvodov, I/O Mapping

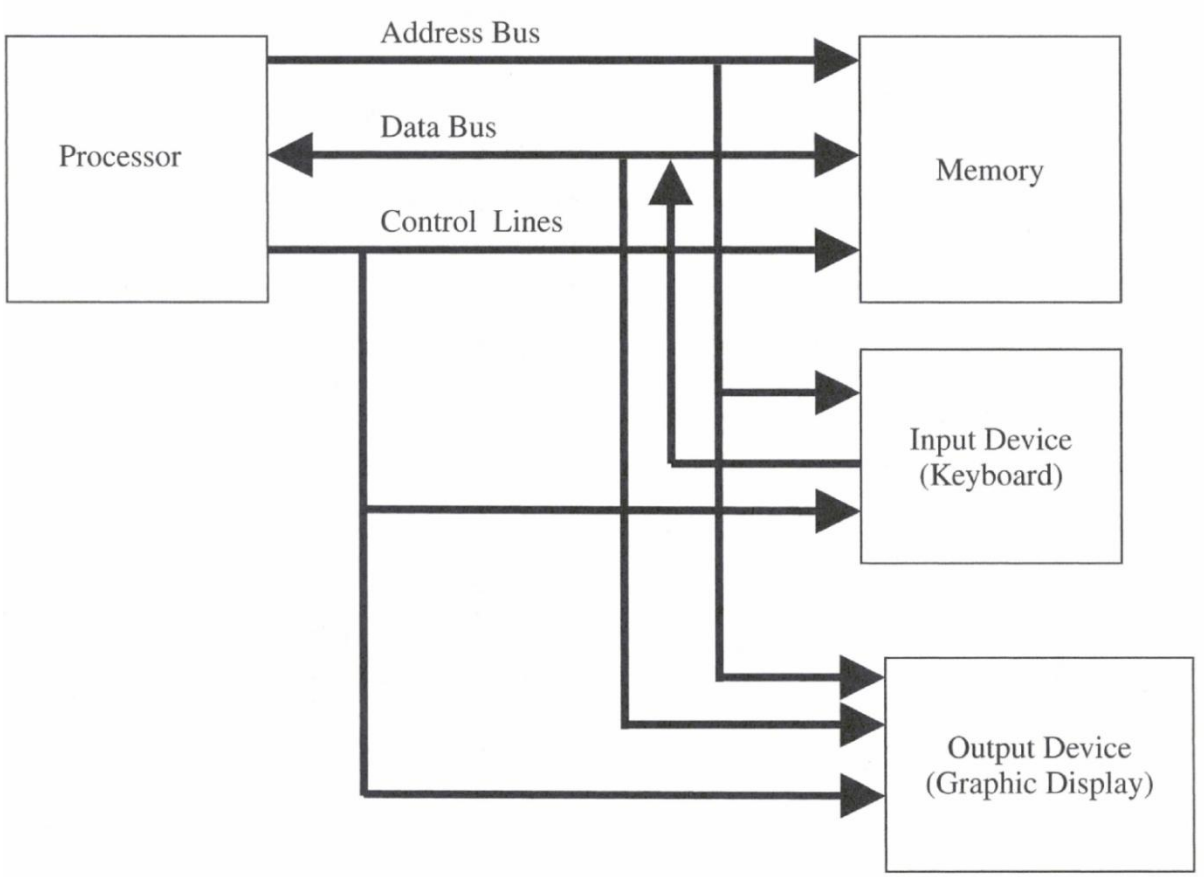
Z uvedeného je zrejmé že procesor musí vedieť príslušné I/O zariadenia adresovať.

- **Pamäťovo mapované I/O** Pre I/O zariadenia možno použiť celú pamäť

– Periférie a pamäť zdieľajú ten istý pamäťový priestor

– Žiadny špeciálny príkaz pre I/O

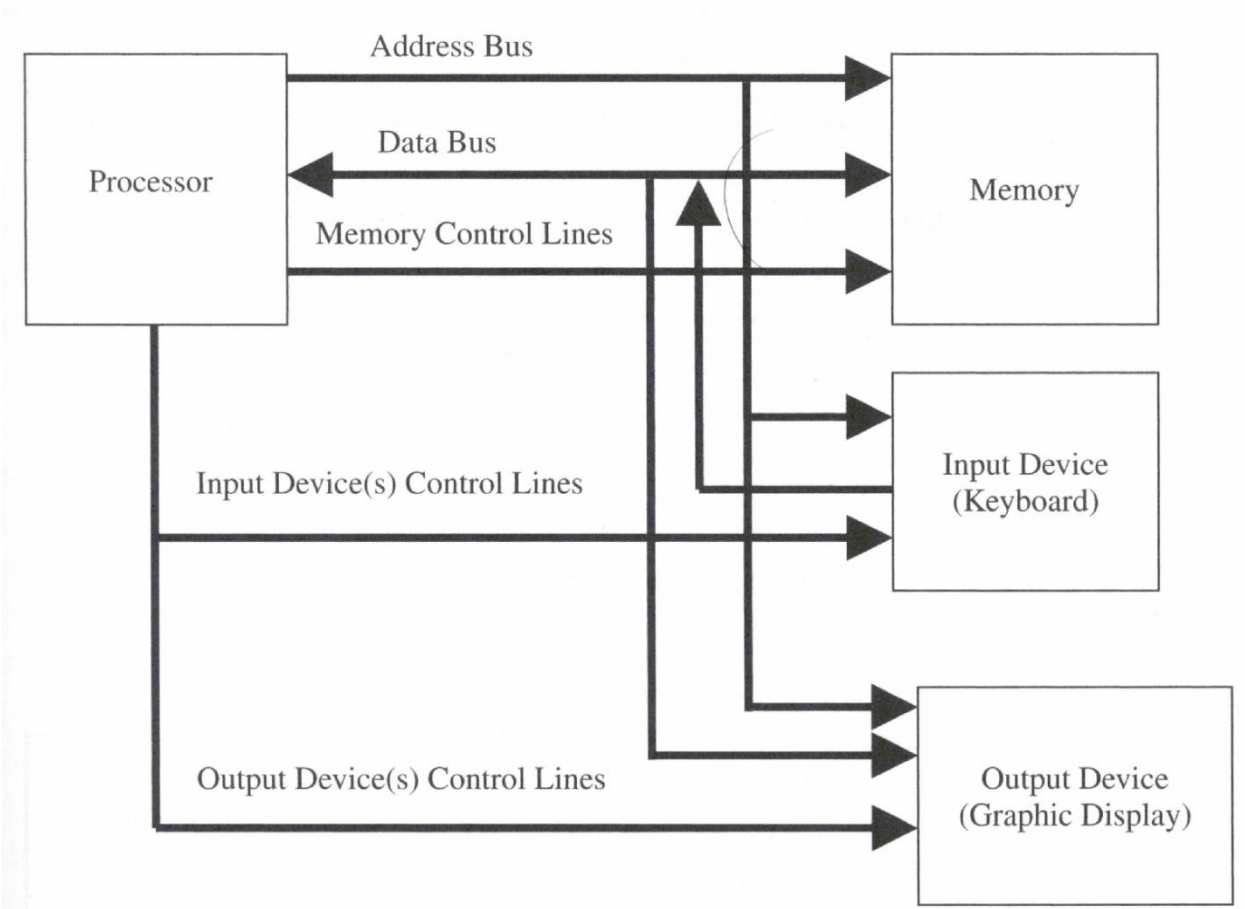
– (Motorola).



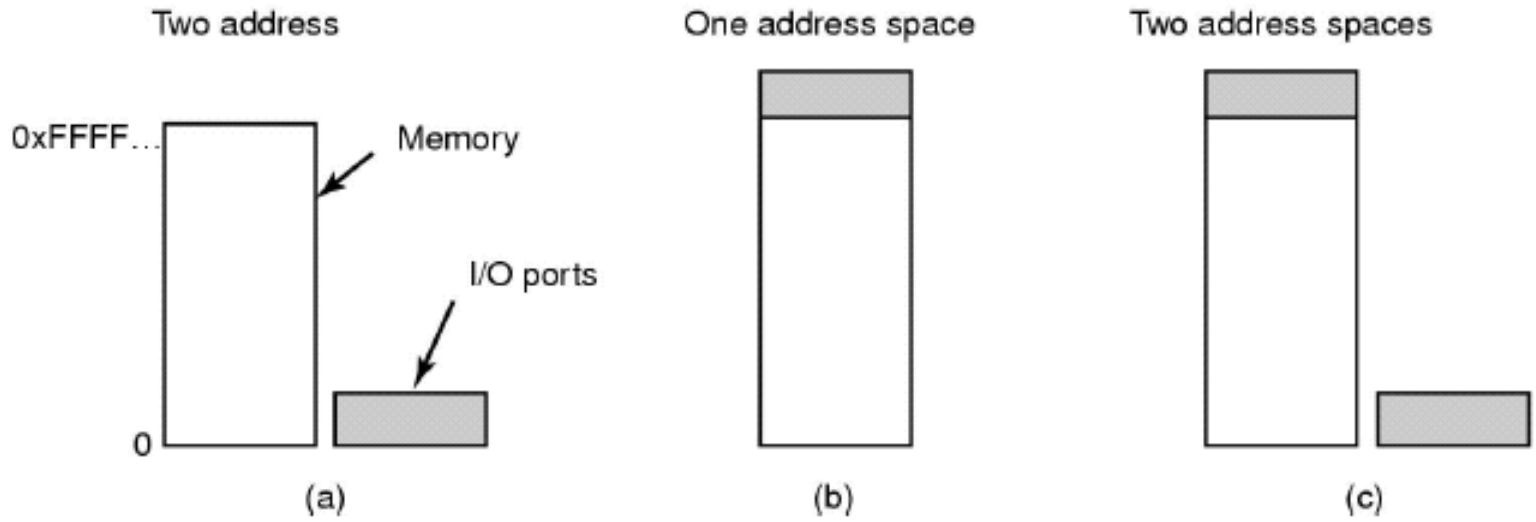
Mapovanie I/O obvodov, I/O Mapping

- **Samostatné mapované I/O zariadenia**

– Oddelený adresný priestor
– Špeciálne vodiče pre výber I/O a pamäť
– Špeciálne príkazy I/O
– (Intel)



Mapovanie I/O obvodov, I/O Mapping



- Separate I/O and memory space PC
- Memory-mapped I/O 8051
- Hybrid AVR

EEPROM – pamäť dát

- kapacita 64B až 8kB
- netreba externé komponenty,
- mimo adresný priestor programu a dát
- Zápis cca 8ms, súčasťou je aj mazanie
- okamžité čítanie
- zapojené ako periféria s GPR registrami
- počet erase/write cyklov 100 000

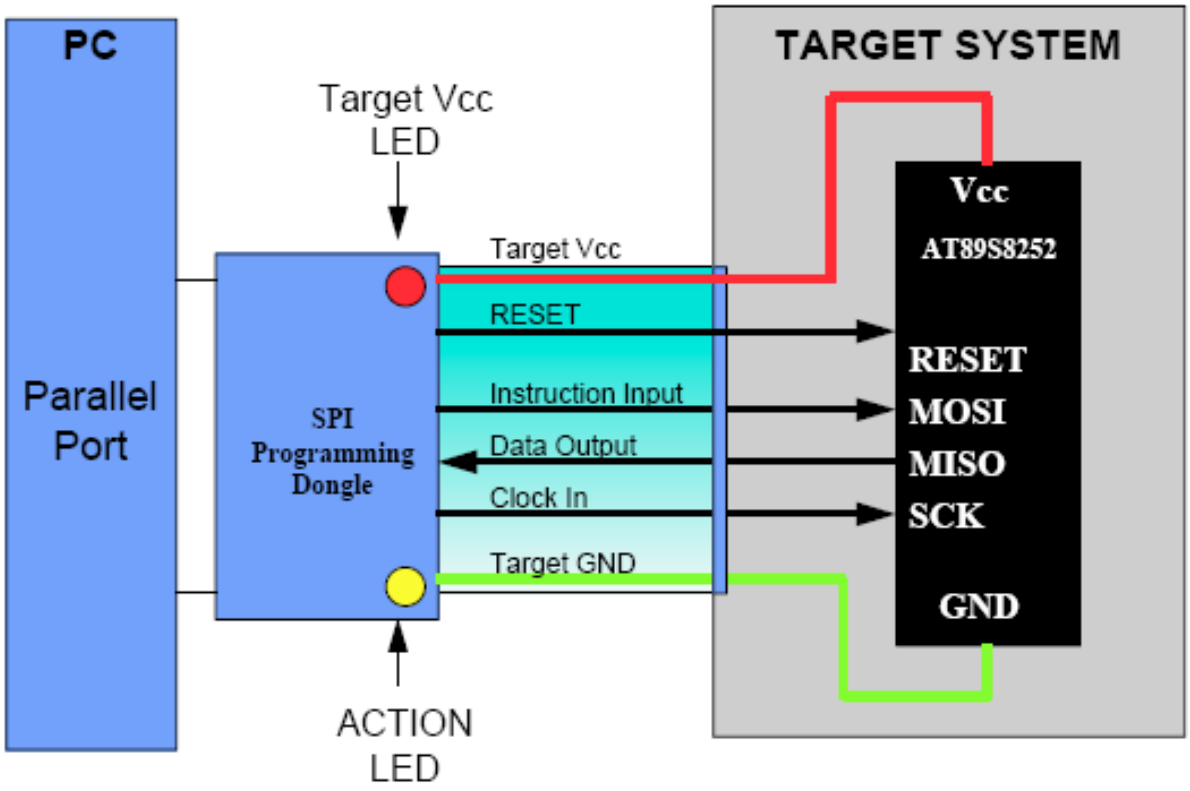
  (1 000 000)

Programovanie:

- Paralelné. Vyžaduje špeciálny programátor. Je rýchle.
- Sériové. Nevyžaduje špeciálny programátor. Môžeme programovať zabudovaný obvod pomocou 6-tich vodičov.

  "**In-System Programming** (**ISP**)".

ISP programátor

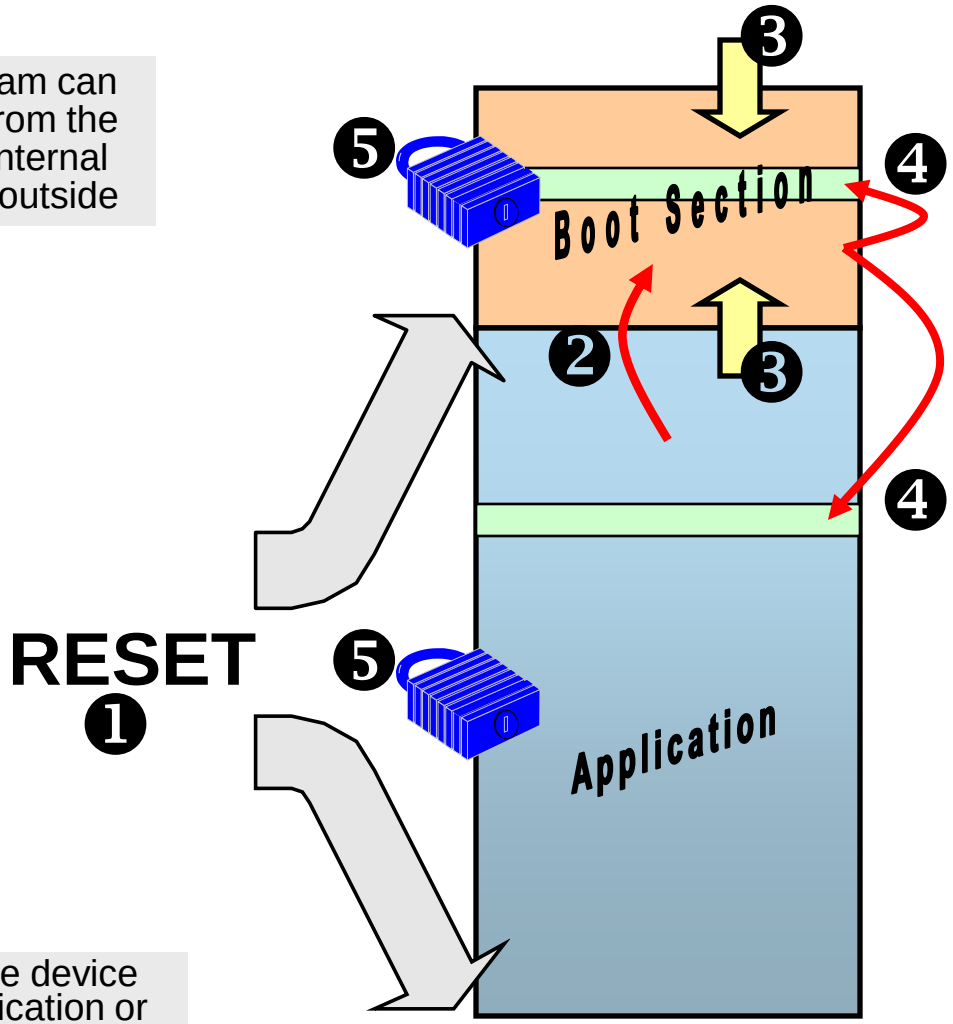


Self Programming Security

❸ Data to program can be obtained from the application, internal RAM or from outside

❷ From the Application, it's possible to jump to the Boot Section

❶ After a Reset the device can start in Application or Boot Loader Mode



❹ The Boot Section can program :

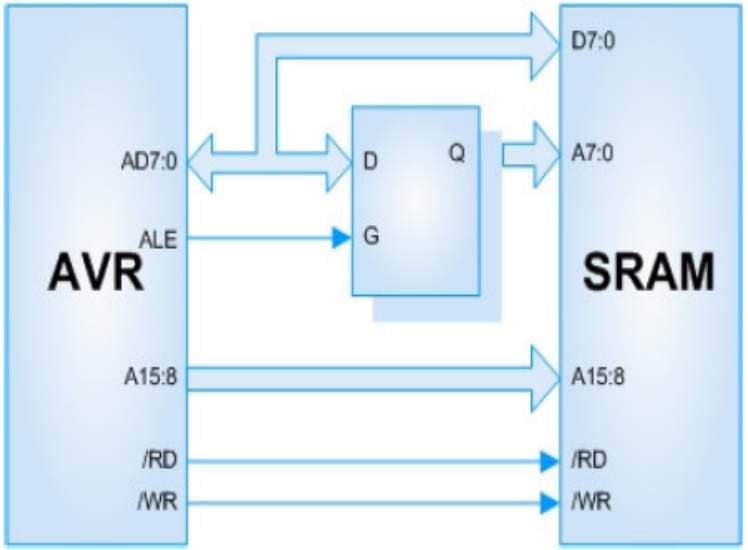
- The Application Section
- The Boot Section

❺ Boot Section and Application Section can be locked to avoid :

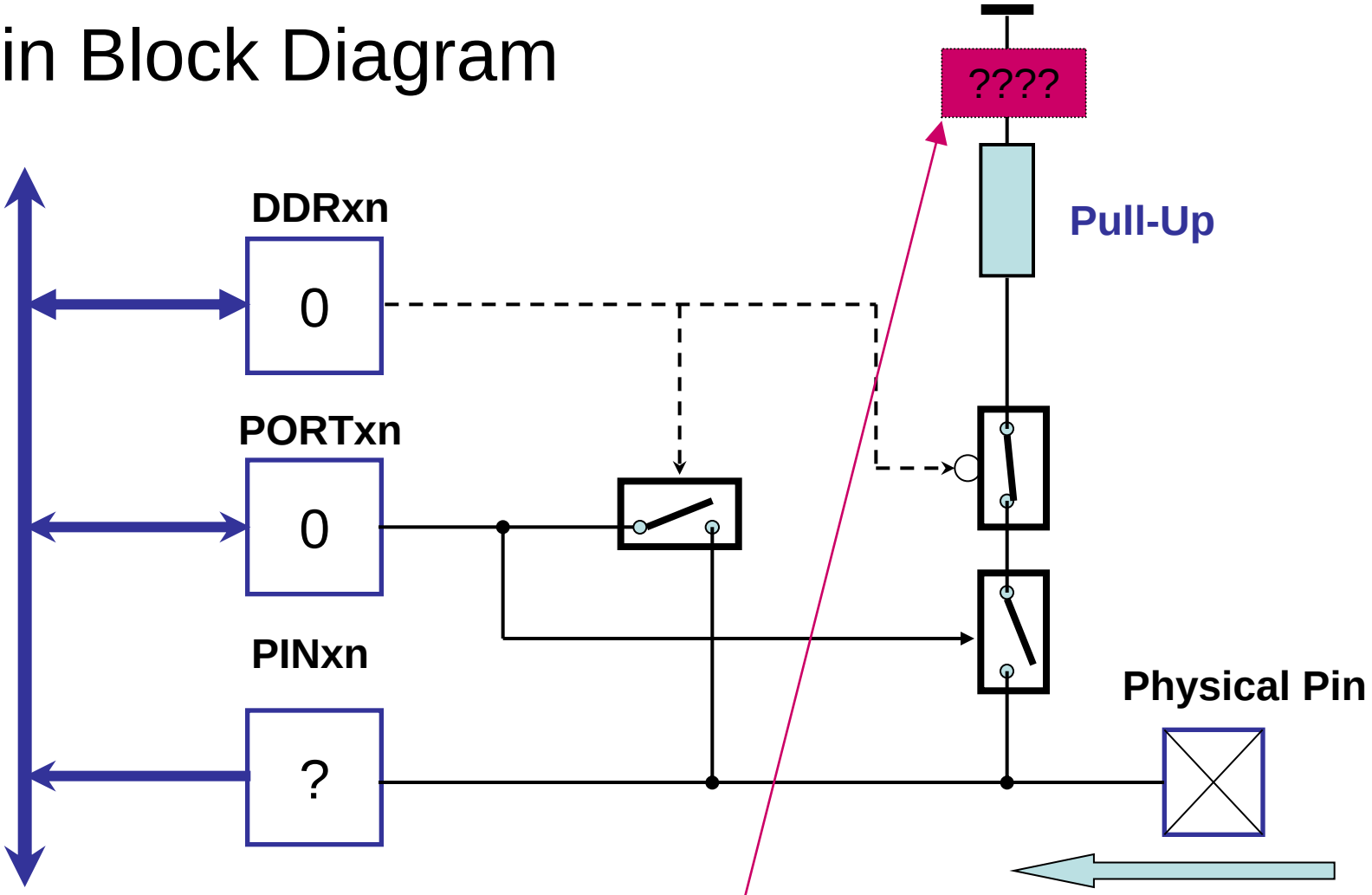
- Read
- Write
- Read and Write

External Memory Interface

- Parallel Interface for external devices or peripherals
- Memory map linear with the internal SRAM
- Parallel bus with 8 data and 16 address lines
  - Extends SRAM memory area up to 64KB
  - Unused high address pins can be turned into general IOs if smaller memories are used
- 4 Wait-state settings
  - Flexible timing settings
  - Independent wait state setting for different external memory sectors
- Integrated Bus-keeper
  - Lower power consumption
- Parts
  - mega256/mega2560
  - mega128/mega64
  - mega162/mega8515

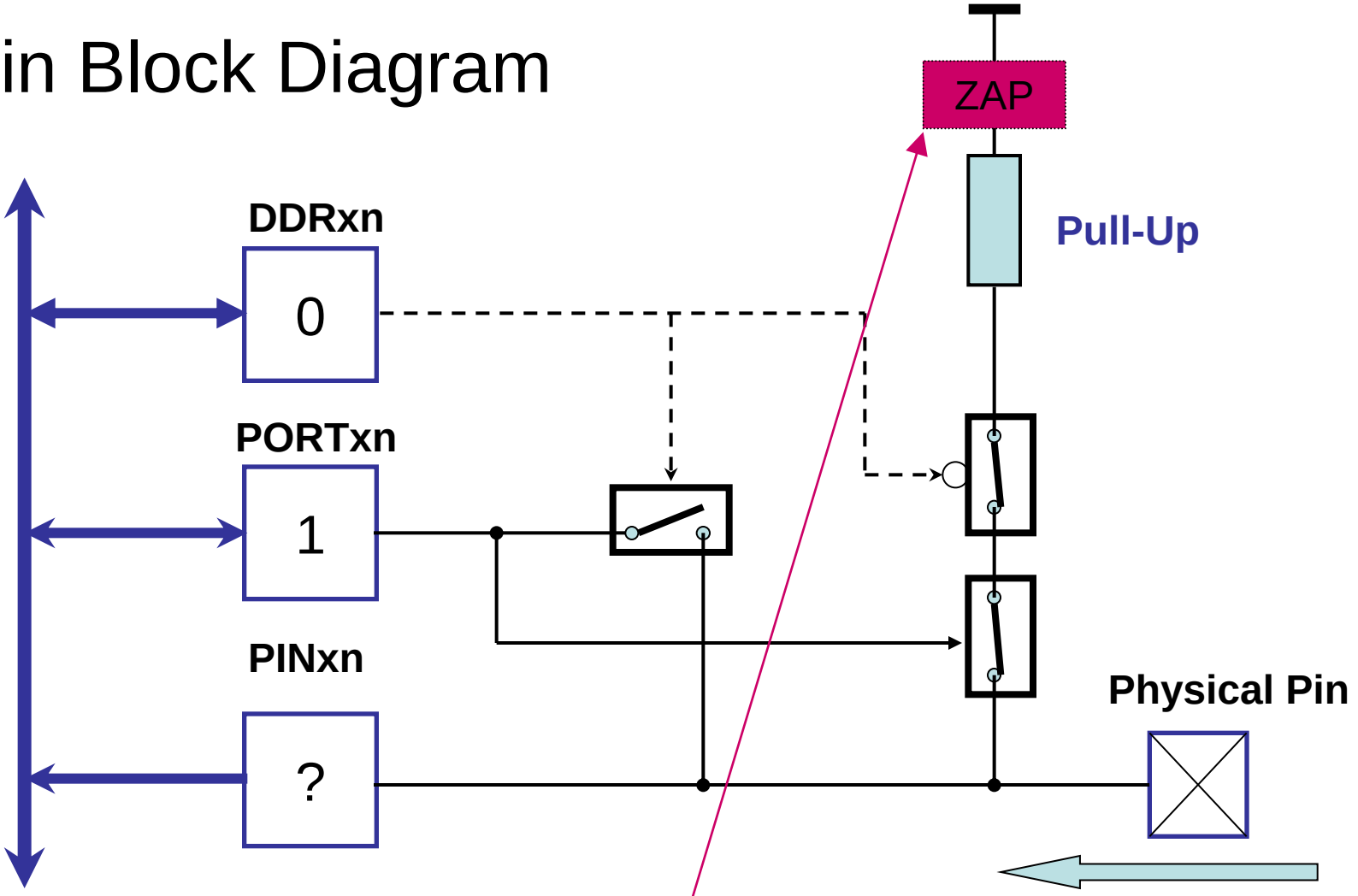


I/O Pin Block Diagram



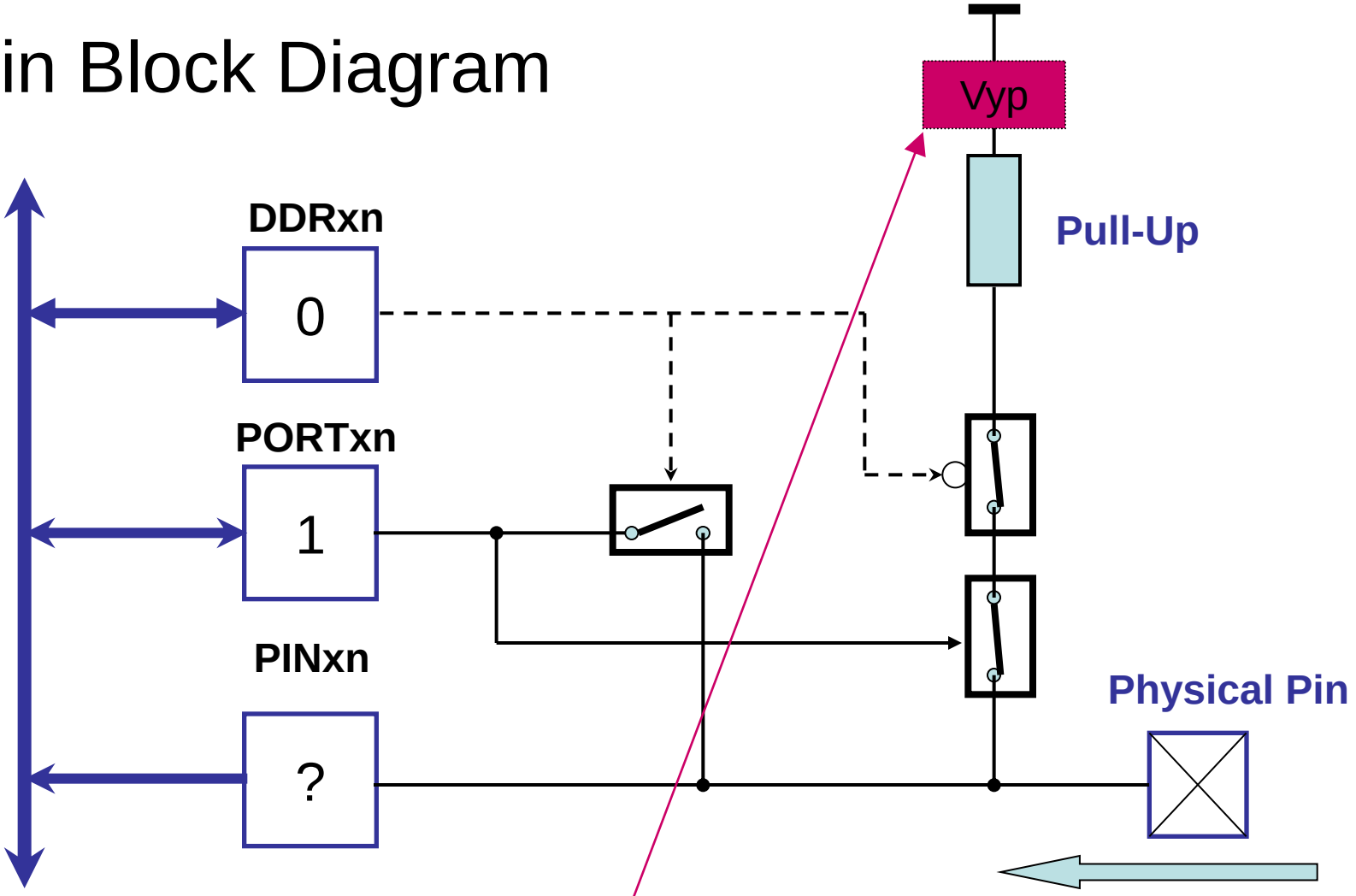
DDRxn	I/O	PORTxn	PUD (SFIOR)	Pull-up	Poznámka
0	In	0	X	No	Tri state (Hi-Z)

I/O Pin Block Diagram



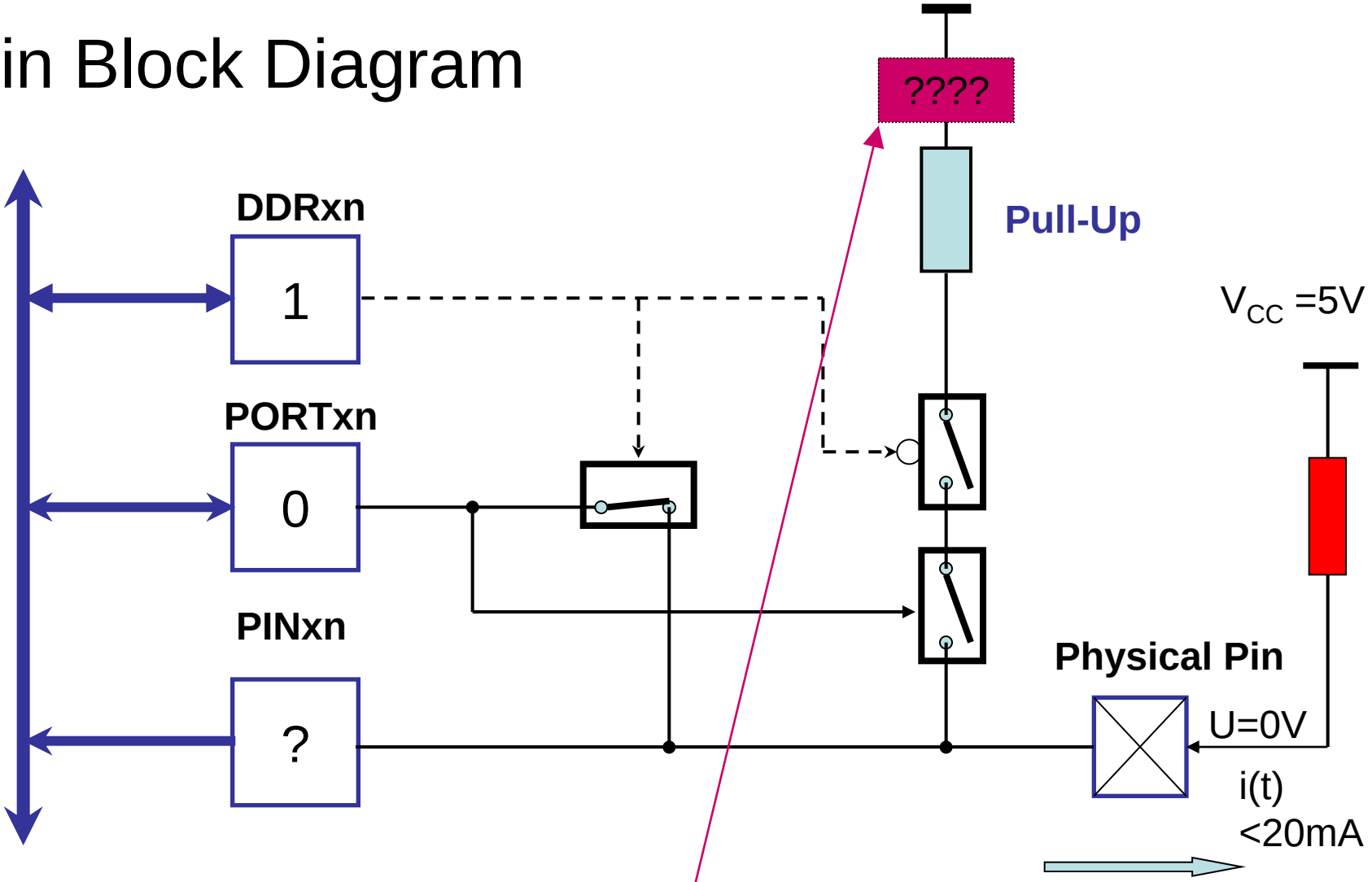
DDRxn	I/O	PORTxn	PUD (SFIOR)	Pull-up	Poznámka
0	In	1	0	Yes	Zdroj prudu (malého), ak „pulled down“

I/O Pin Block Diagram



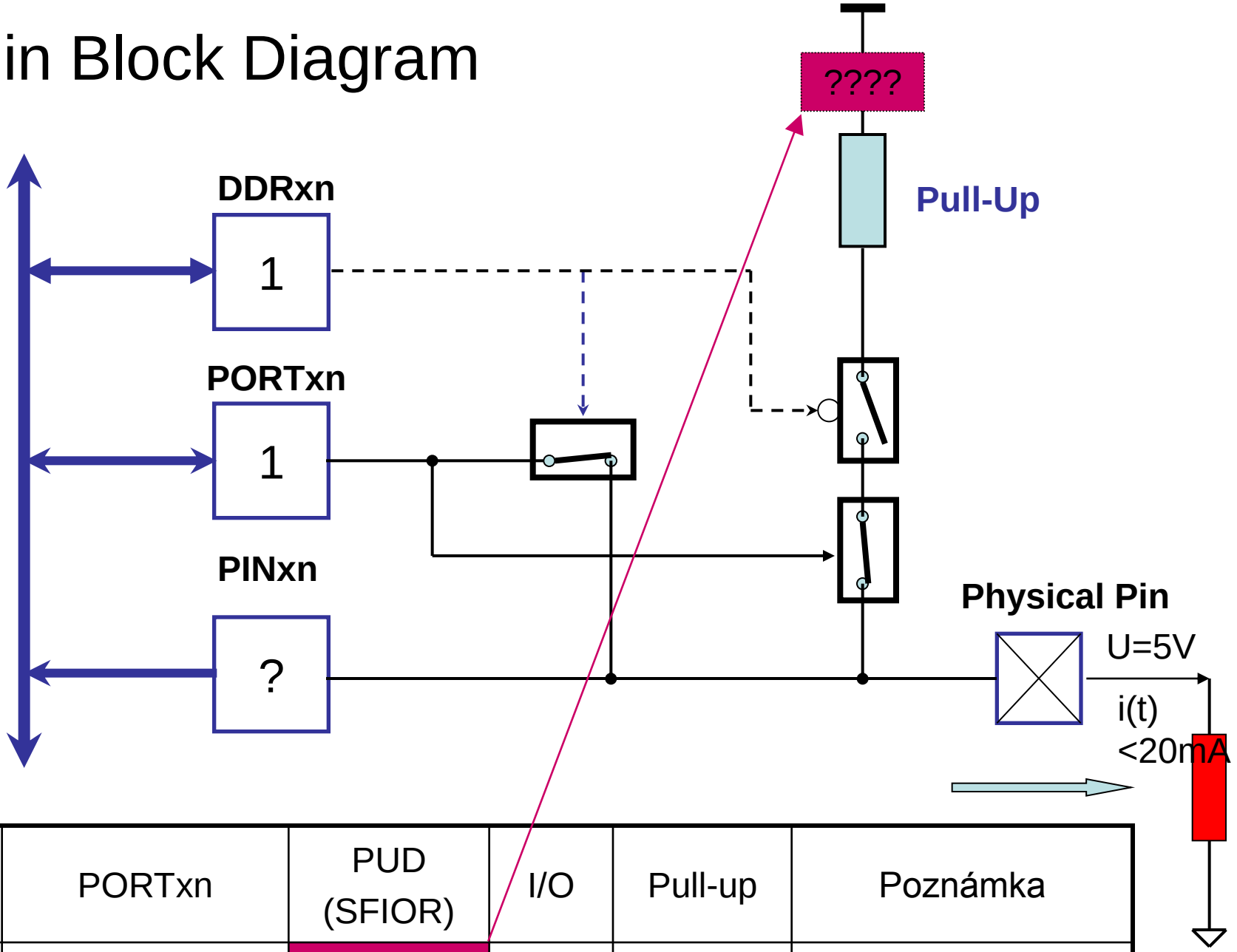
DDRxn	I/O	PORTxn	PUD (SFIOR)	Pull-up	Poznámka
0	In	1	1	No	Tri state (Hi-Z)

I/O Pin Block Diagram



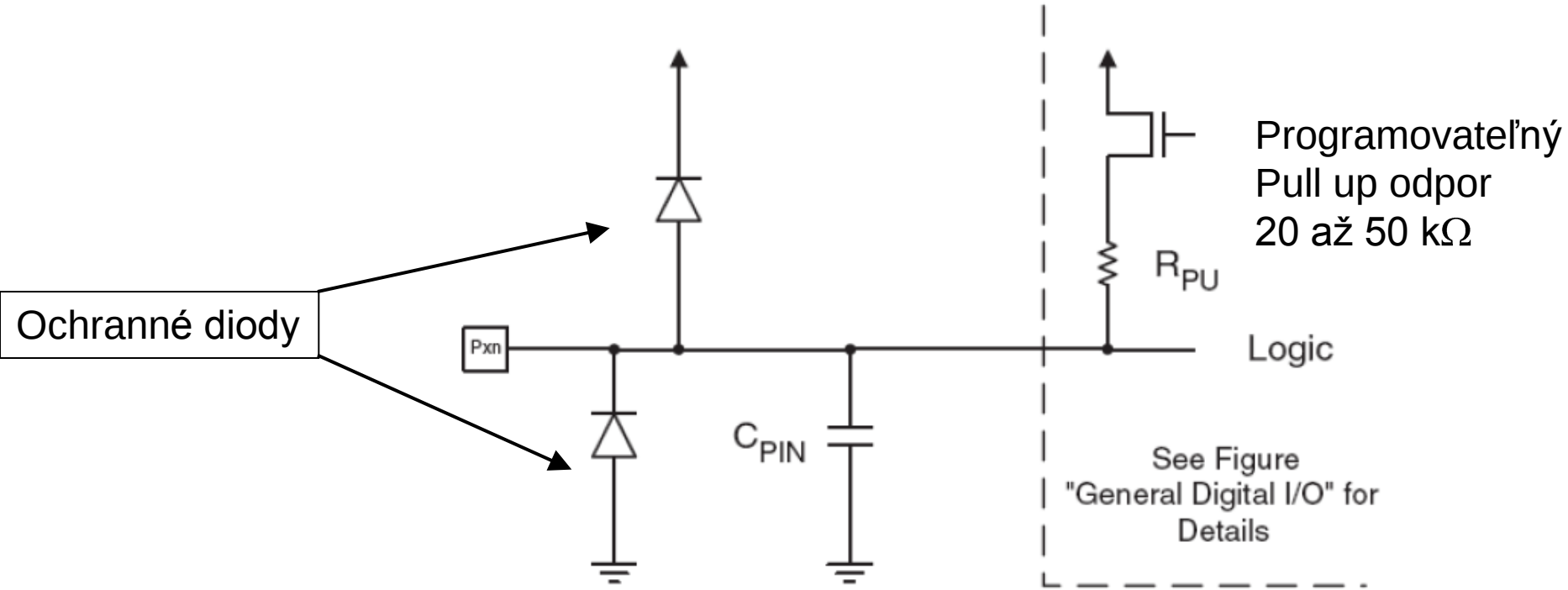
DDRxn	I/O	PORTxn	PUD (SFIOR)	Pull-up	Poznámka
1	Out	0	X	No	Out - L (Sink)

I/O Pin Block Diagram



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

I/O Port – zapojenie vstupného pinu



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

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

I/O Port – architektúra portu

SFIOR

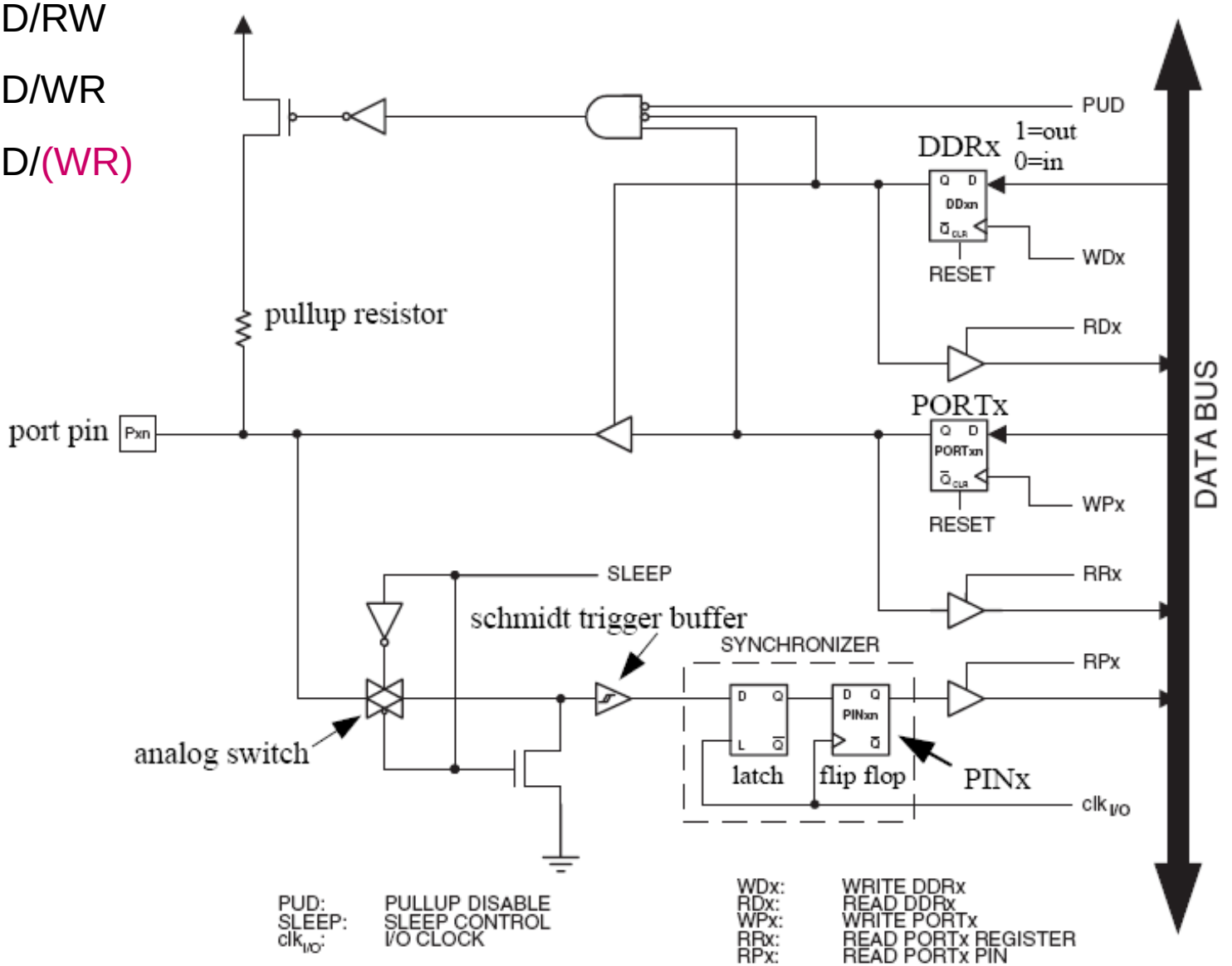
						PUD		

DDRx RD/RW

PORTx RD/WR

PINx RD/(WR)

x = A,B,.. F



Programovanie I/O bit-ov

XOR:
PORTB = PORTB ^ 0x20 // invertovanie, prepnutie, bitu **b5**
PORTB ^ = 0x20

OR:
PORTB = PORTB | 0x12 // nastavenie bitov **b4** a **b1** do log.1
PORTB |= (1 << PB4)|(1 << PB1)

AND:
PORTB = PORTB & 0xFC // nastavenie bitov **b1** a **b0** do log.0
PORTB &= ~((1 << PB1)|(1 << PB0))

DDRB	b7	b6	b5	b4	b3	b2	b1	b0
	0	0	0	0	0	0	1	1

IN

OUT

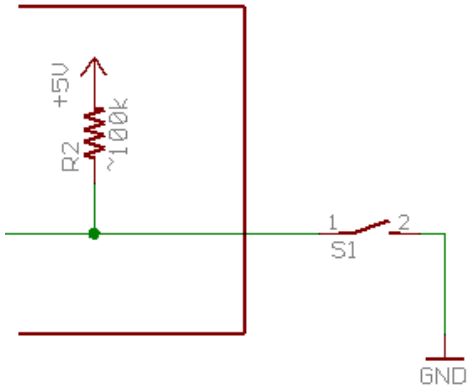
PORTB	b7	b6	b5	b4	b3	b2	b1	b0
	0	0	0	0	1	0	0	1

PULLUP

PINB	b7	b6	b5	b4	b3	b2	b1	b0
	?	?	?	?	?	?	0	1

I/O Príklad: jednoduchý kontakt

Netreba pripájať žiadne externé súčiastky



High Level Languages

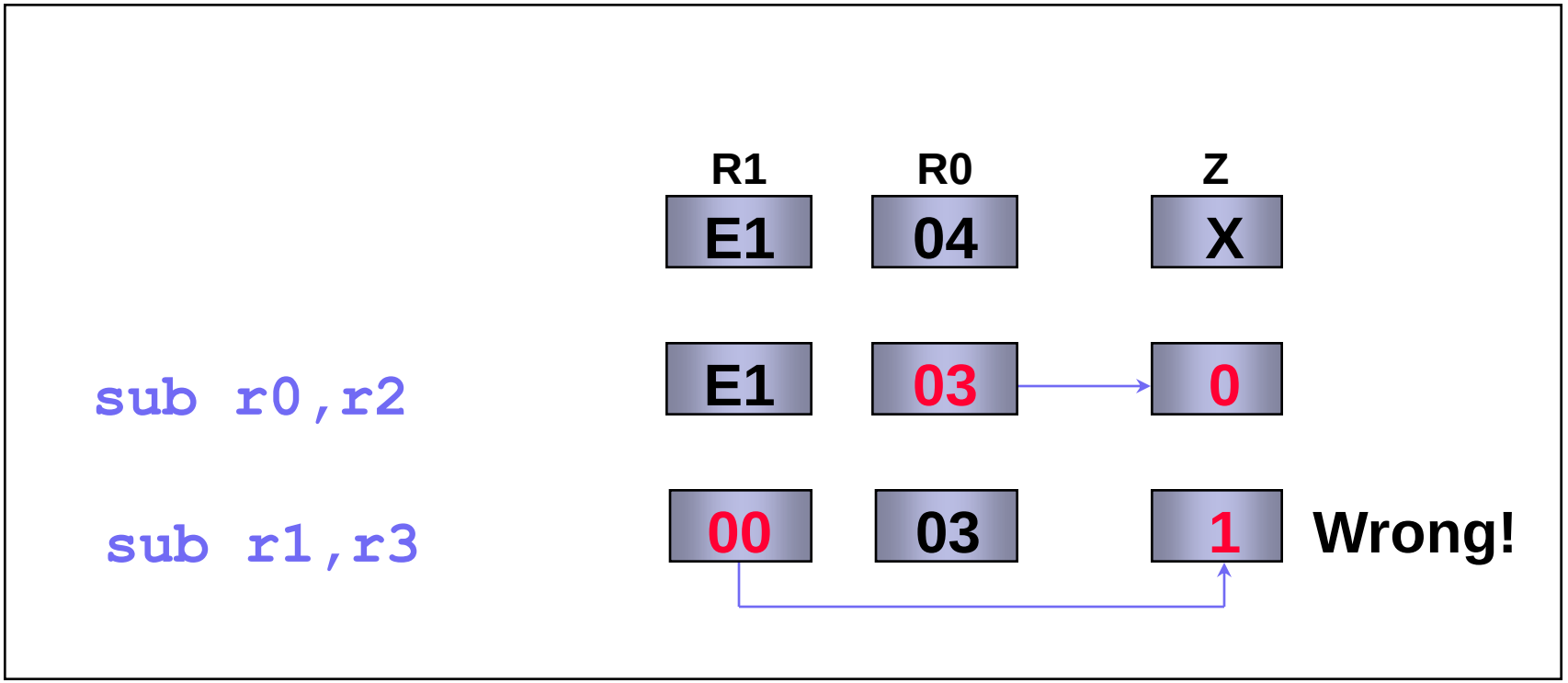
HLL (C-ko) bolo pre 8-bitové microcotrollery menej výhodné ako assembler.
$\Rightarrow$ ATMEL riešil tento problém tak, že Hardware a inštrukčná sada sa „prispôsobili“ HLL.

- RISC procesor (počet inštrukcií nie je redukovaný za každú cenu, viď. ďalší príklad)
- 32 GPR (akumulátorov)

Rozdieľ dvoch 16-Bit-ových čísiel

Without Zero Flag Propagation

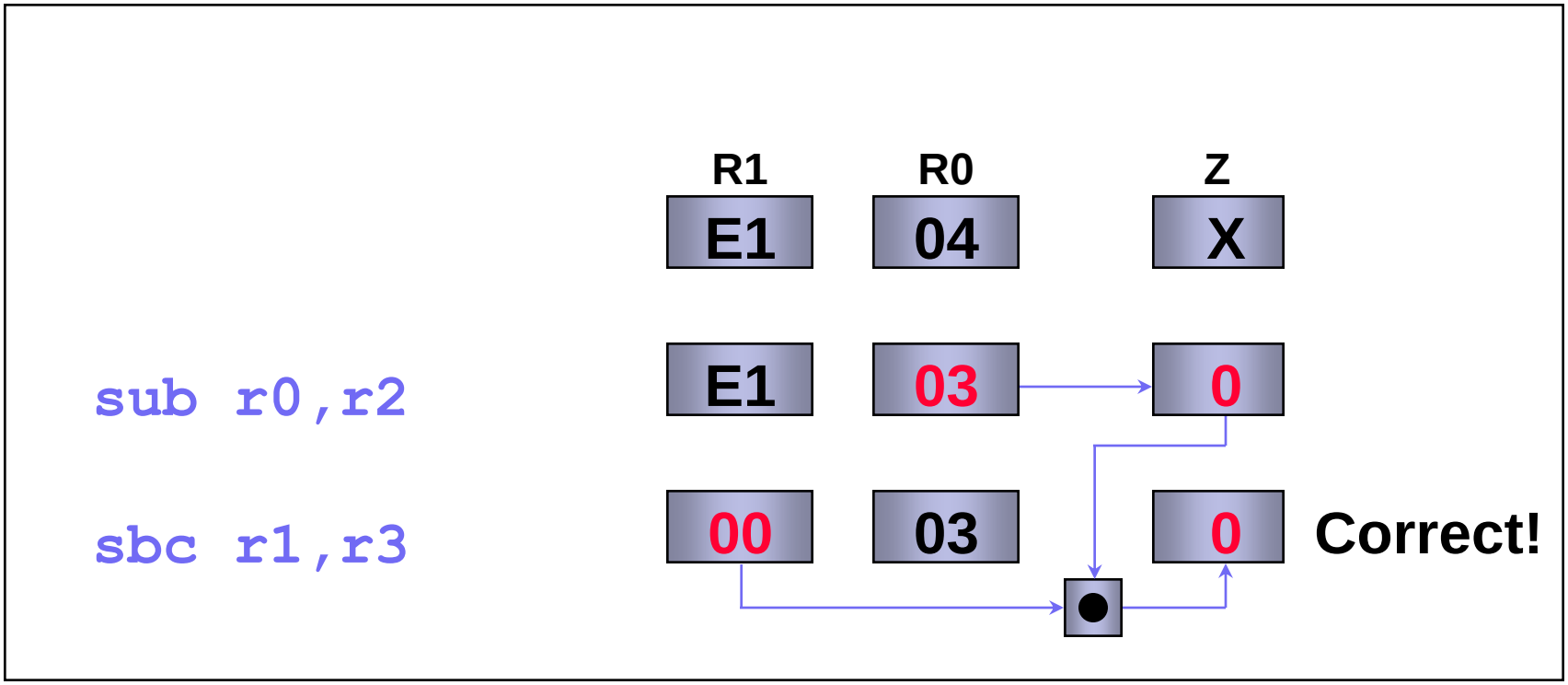
R1:R0 - R3:R2 ($E104 - $E101)



Rozdieľ dvoch 16-Bit-ových čísiel

With Zero Flag Propagation

R1:R0 - R3:R2 ($E104 - $E101)



Na druhej strane uviesť, že niekedy vedeli čo bude mikrokontroler robiť:
Súčasťou inštrukčnej sady sú:

LSL	Rd	Logical Shift Left	Rd(n+1)←Rd(n),Rd(0)←0,C←Rd(7)	Z,C,N,V,H	1
LSR	Rd	Logical Shift Right	Rd(n)←Rd(n+1),Rd(7)←0,C←Rd(0)	Z,C,N,V	1
ROL	Rd	Rotate Left Through Carry	Rd(0)←C,Rd(n+1)←Rd(n),C←Rd(7)	Z,C,N,V,H	1
ROR	Rd	Rotate Right Through Carry	Rd(7)←C,Rd(n)←Rd(n+1),C←Rd(0)	Z,C,N,V	1
ASR	Rd	Arithmetic Shift Right	Rd(n) ← Rd(n+1), n=0..6	Z,C,N,V	1

Na prvý pohľad sa zdá, že „chýba“ ASL.

Problémom je zobrazovanie a operácie so záporným číslami.
MSB predstavuje znamienko zobrazovaného čísla.
Do 8-bitového čísla vieme v dvojkovom doplnkovom kóde zobraziť čísla <-128, 127>.
Posun o jedno miesto doľava znamená násobenie dvoma.
Posun o jedno miesto doprava znamená delenie dvoma.
Číslo -64_{10} = 0xC0 = b1100 0000
Číslo -65_{10} = 0xBF = b1011 1111

$-64_{10} * 2_{10} = 0x80 = b1000\ 0000 = -128_{10}$ Zobraziteľné číslo, nenastalo pretečenie
$-65_{10} * 2_{10} = 0x7E = b0111\ 1110 = +126_{10}$ Nezobraziteľné číslo, nastalo pretečenie
„Modulo aritmetika“ $-65_{10} * 2_{10} = -130_{10} + 256_{10} = +126_{10}$