Mikropočítačové Systémy - MIPS

Distribuované vnorené počítačové systémy

Distributed Embedded Computer System

(Microcontrollers - MCU)

Monolitické mikropočítače - MMP

Prednáška 1.
Úvod.

http://senzor.robotika.sk => MIPS

- Prednášky:
  - Chamraz Štefan: (D-107, stefan.chamraz@stuba.sk)
  - Balogh Richard, (D-109, richard.balogh@stuba.sk)
  - termín: Pondelok 13:00 - 15:00

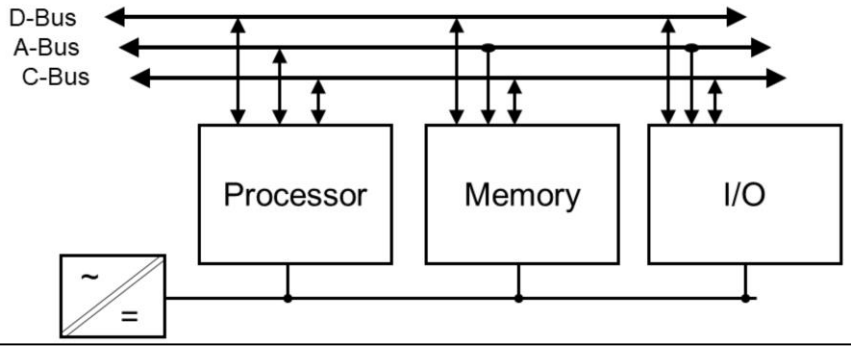
- Cvičenia:
  - Balogh Richard, (D-109, richard.balogh@stuba.sk)
  - Chamraz Štefan, (D-107, stefan.chamraz@stuba.sk)
  - Matej Rábek (matej.rabek@stuba.sk)
    - Štvrtok: 08:00 - 10:00, Balogh | 10:00 – 12:00, Chamraz | 12:00 – 14:00, Chamraz | 14:00 – 16:00 Rábek

– **Podmienky na absolvovanie predmetu:**
– **Cvičenia: X** bodov. Výsledky priebežného hodnotenia sú súčasťou záverečného hodnotenia.
– **Skúška: 100 - X** bodov

Literatúra: Katalógové listy – KL. Prednášky, cvičenia.
https://meet.google.com/eaf-uggp-skd
http://senzor.robotika.sk/mips

Štruktúra počítača

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



3

Už na predmete **Základy počítačov** sme si povedali, že počítač sa skladá z troch základných častí: procesor (CPU), pamäť a periferie.

Už niekoľko 10-ročí sa vyrábajú malé počítače integrované v jednom púzdre, ktoré tvoria jeden celok - monolit. =>názov - monolitické počítače (mikropočítače).

V praxi sa stretneme aj s inými názvami: mikročip, mikrokontroler a mikroprocesor. Niektoré sa pokúsim vysvetliť:

Mikrokontroler je vlastne malý počítač na jednom čipe. Mikrokontroler je procesor s pamäťou a ostatnými perifériami ako sú porty, C/T, USART, I2C, prerušovací podsystém, A/D (D/A) prevodník s mutiplexorom, ... v jednom púzdre. Programuje sa podobne ako počítač. Spracováva vstupné dáta, realizuje výpočty a generuje výstupy. Mikrokontrolery sú súčasťou tzv. „Embedded systems“

Procesor má väčšinou vyvedené na piny adresnú a dátovú zbernicu a piny riadiacej zbernice so signálmi ako RD/WR, INT, ...

Mikroprocesor nemôže pracovať samostatne, potrebuje pripojiť externé obvody.

Na druhej strane mikrokontroler nemá priamo vyvedenú adresnú a dátovú

zbernicu ale má veľa univerzálnych I/O pinov. Napr.: PA0-PA7, PB0-PB7, ... ktoré môžu mať aj iné funkcie ako I/O piny. Napr. analógové vstupy.

My budeme často používať názov monolitické mikropočítače a skratku MMP ako náhradu za mikrokontroler. K tomuto pojmu používa anglická literatúra skratku MCU. V KL ATMEL-u táto skratka nie je zavedená. Len sa používa. Napr. MCUCR označuje riadiaci register mikrokontrolera.

Podstatné nie je pomenovanie, ale čo pod tým pojmom rozumieme. Výrobcovia radi používajú nadnesené pomenovania. A našou úlohou je sa v tom vyznať.

Embedded System: Niečo podobné pojmu **Mechatronika**. Výsledný systém vznikne spojením mechanických, elektrických, ... častí, ktoré sú ovládané zabudovaným počítačom. Celé je to koncipované tak, aby to plnilo nejaký účel.

Real-Time System: Tu si povieme najjednoduchšiu definíciu systému reálneho času. Správny výsledok v správnom (požadovanom) čase. Existujú OS reálneho času.

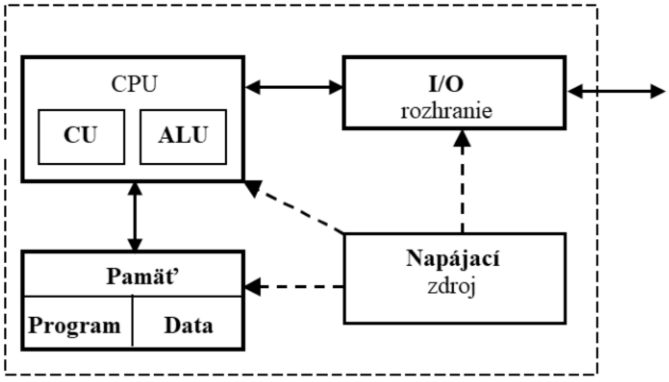
Von Neumannov počítač

Princetonská architektúra

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.



Požadujeme: *rýchly procesor a veľkú pamäť* =>

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

4

Počítač je vo svojej podstate sériovo pracujúci stroj. A od jeho vzniku (druhá polovica minulého storočia) sa „všetci“ snažia o zvýšenie výpočtového výkonu:

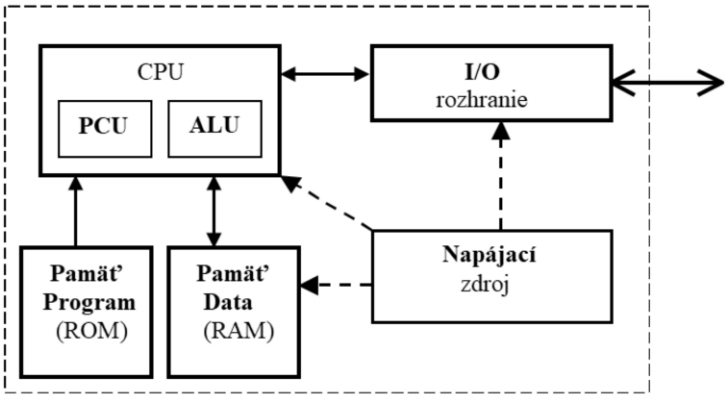
- Jeden zo spôsobov je zvýšenie rýchlosti výpočtov, zvýšenie taktovacej frekvencie procesora. Činnosť počítača je synchronizovaná tzv. hodinovým signálom – oscilátorom. Rýchlosť vykonávania operácií počítača sa meria v bezrozmerných veličinách strojových cykloch (SC).
- Druhým spôsob je sparalelnenie v ňom prebiehajúcich procesov:
    - Zreťazenie inštrukcií.
    - Delegovanie právomocí – vytvorenie periférií.
        - „8278“. Obsluha klávesnice.
        - „8253“. Čítače/časovače.
        - „8255“. Binárne vstupy a výstupy.
        - „8259“. Obsluha prerušení.
        - „8251“. USART.
        - ...

Poznáme dve základné koncepcie počíitačov: von Neumannovu a harvardskú. Prvá je pomenovaná po američanovi maďarského pôvodu (spolupracoval s

Goldstine-om). Prevzali základné princípy ENIAC-u a navrhli ukladať program do pamäte. Dá sa to povedať aj takto: uvedomili si nedostatky ENIACU: *Počítač nie je jednoúčelový stroj, a dá sa programovať aj ináč ako prestavením prepínačov, ...ENIAC sa programoval tak, že sa pospájali body drôtikmi*. Obdobný princíp činnosti počítacieho stroja navrhol o vyše sto rokov skorej Charles Babbage.

Vo von Neumannovej architektúre sú program a dáta uložené spolu v jednom adresnom priestore a pristupujeme k nim cez tú istú zbernicu. To môže spôsobiť problém – oneskorenie. Niekedy sa tento problém nazýva: von Neumann bottleneck.

Harvardská architektúra (Aiken)



5

V MMP sa najčastejšie požíva harvardska architektúra, kvôli tomu, že je oddelená pamäť dát od pamäti programu. To umožnuje vyrobiť každú pamäť inou technológiou a o inej šírke ukladaných dát. Táto koncepcia počítača je postavená na základoch Babbage-vých prác. Kolektív vedcov viedol Howard Aiken, ale koncepcia je pomenovaná po univerzite.

Pamäť dát býva zvyčajne organizovaná po bajtoch, resp. slovách. Oddelenie pamäťových priestorov umožňuje prispôsobiť šírku pamäte k strojovej inštrukcii (napr. 12, 14, 16, 24, ... bitov).

Ako malú nevýhodu môžeme uviesť, že táto architektúra je hardwarovo zložitejšia, pretože vyžaduje dve zbernice, dve samostatné pamäte a súčasný prístup k obom pamätiam.

Ďalšie delenie počítačov je možné podľa použiteho procesora: CISC alebo RISC. MMP sa vyrábajú s architektúrou RISC, aj keď veľa krát dosť zjednodušenou.

CISC:

Table 16. M68HC05 Instruction Set Opcode Map

	Bit Manipulation		Branch	Read-Modify-Write					Control		Register/Memory					
	DIR	DIR	REL	DIR	INH	INH	IX1	IX	INH	INH	IMM	DIR	EXT	IX2	IX1	IX
MSB / LSB	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	BRSET0 5 3 DIR	BSET0 5 2 DIR	BRA 3 2 REL	NEG 5 2 DIR	NEGA 3 1 INH	NEGX 3 1 INH	NEG 6 2 IX1	NEG 5 1 IX	RTI 9 1 INH		SUB 2 2 IMM	SUB 3 2 DIR	SUB 4 3 EXT	SUB 5 3 IX2	SUB 4 2 IX1	SUB 3 1 IX
1	BRCLR0 5 3 DIR	BCLR0 5 2 DIR	BRN 3 2 REL						RTS 6 1 INH		CMP 2 2 IMM	CMP 3 2 DIR	CMP 4 3 EXT	CMP 5 3 IX2	CMP 4 2 IX1	CMP 3 1 IX
2	BRSET1 5 3 DIR	BSET1 5 2 DIR	BHI 3 2 REL		MUL 11 1 INH						SBC 2 2 IMM	SBC 3 2 DIR	SBC 4 3 EXT	SBC 5 3 IX2	SBC 4 2 IX1	SBC 3 1 IX
3	BRCLR1 5 3 DIR	BCLR1 5 2 DIR	BLS 3 2 REL	COM 5 2 DIR	COMA 3 1 INH	COMX 3 1 INH	COM 6 2 IX1	COM 5 1 IX	SWI 10 1 INH		CPX 2 2 IMM	CPX 3 2 DIR	CPX 4 3 EXT	CPX 5 3 IX2	CPX 4 2 IX1	CPX 3 1 IX
4	BRSET2 5 3 DIR	BSET2 5 2 DIR	BCC 3 2 REL	LSR 5 2 DIR	LSRA 3 1 INH	LSRX 3 1 INH	LSR 6 2 IX1	LSR 5 1 IX			AND 2 2 IMM	AND 3 2 DIR	AND 4 3 EXT	AND 5 3 IX2	AND 4 2 IX1	AND 3 1 IX
5	BRCLR2 5 3 DIR	BCLR2 5 2 DIR	BCS/BLO 3 2 REL								BIT 2 2 IMM	BIT 3 2 DIR	BIT 4 3 EXT	BIT 5 3 IX2	BIT 4 2 IX1	BIT 3 1 IX
6	BRSET3 5 3 DIR	BSET3 5 2 DIR	BNE 3 2 REL	ROR 5 2 DIR	RORA 3 1 INH	RORX 3 1 INH	ROR 6 2 IX1	ROR 5 1 IX			LDA 2 2 IMM	LDA 3 2 DIR	LDA 4 3 EXT	LDA 5 3 IX2	LDA 4 2 IX1	LDA 3 1 IX
7	BRCLR3 5 3 DIR	BCLR3 5 2 DIR	BEQ 3 2 REL	ASR 5 2 DIR	ASRA 3 1 INH	ASRX 3 1 INH	ASR 6 2 IX1	ASR 5 1 IX		TAX 2 1 INH		STA 4 2 DIR	STA 5 3 EXT	STA 6 3 IX2	STA 5 2 IX1	STA 4 1 IX
8	BRSET4 5 3 DIR	BSET4 5 2 DIR	BHCC 3 2 REL	ASL/LSL 5 2 DIR	ASLA/LSLA 3 1 INH	ASLX/LSLX 3 1 INH	ASL/LSL 6 2 IX1	ASL/LSL 5 1 IX		CLC 2 1 INH	EOR 2 2 IMM	EOR 3 2 DIR	EOR 4 3 EXT	EOR 5 3 IX2	EOR 4 2 IX1	EOR 3 1 IX
9	BRCLR4 5 3 DIR	BCLR4 5 2 DIR	BHCS 3 2 REL	ROL 5 2 DIR	ROLA 3 1 INH	ROLX 3 1 INH	ROL 6 2 IX1	ROL 5 1 IX		SEC 2 1 INH	ADC 2 2 IMM	ADC 3 2 DIR	ADC 4 3 EXT	ADC 5 3 IX2	ADC 4 2 IX1	ADC 3 1 IX
A	BRSET5 5 3 DIR	BSET5 5 2 DIR	BPL 3 2 REL	DEC 5 2 DIR	DECA 3 1 INH	DECX 3 1 INH	DEC 6 2 IX1	DEC 5 1 IX		CLI 2 1 INH	ORA 2 2 IMM	ORA 3 2 DIR	ORA 4 3 EXT	ORA 5 3 IX2	ORA 4 2 IX1	ORA 3 1 IX
B	BRCLR5 5 3 DIR	BCLR5 5 2 DIR	BMI 3 2 REL							SEI 2 1 INH	ADD 2 2 IMM	ADD 3 2 DIR	ADD 4 3 EXT	ADD 5 3 IX2	ADD 4 2 IX1	ADD 3 1 IX
C	BRSET6 5 3 DIR	BSET6 5 2 DIR	BMC 3 2 REL	INC 5 2 DIR	INCA 3 1 INH	INCX 3 1 INH	INC 6 2 IX1	INC 5 1 IX		RSP 2 1 INH		JMP 2 2 DIR	JMP 3 3 EXT	JMP 4 3 IX2	JMP 3 2 IX1	JMP 2 1 IX
D	BRCLR6 5 3 DIR	BCLR6 5 2 DIR	BMS 3 2 REL	TST 4 2 DIR	TSTA 3 1 INH	TSTX 3 1 INH	TST 5 2 IX1	TST 4 1 IX		NOP 2 1 INH	BSR 6 2 REL	JSR 5 2 DIR	JSR 6 3 EXT	JSR 7 3 IX2	JSR 6 2 IX1	JSR 5 1 IX
E	BRSET7 5 3 DIR	BSET7 5 2 DIR	BIL 3 2 REL						STOP 2 1 INH		LDX 2 2 IMM	LDX 3 2 DIR	LDX 4 3 EXT	LDX 5 3 IX2	LDX 4 2 IX1	LDX 3 1 IX
F	BRCLR7 5 3 DIR	BCLR7 5 2 DIR	BIH 3 2 REL	CLR 5 2 DIR	CLRA 3 1 INH	CLRX 3 1 INH	CLR 6 2 IX1	CLR 5 1 IX	WAIT 2 1 INH	TXA 2 1 INH		STX 4 2 DIR	STX 5 3 EXT	STX 6 3 IX2	STX 5 2 IX1	STX 4 1 IX

INH = Inherent
IMM = Immediate
DIR = Direct
EXT = Extended
REL = Relative
IX = Indexed, No Offset
IX1 = Indexed, 8-Bit Offset
IX2 = Indexed, 16-Bit Offset

LSB of Opcode in Hexadecimal

MSB / LSB: 0 / 0 — MSB of Opcode in Hexadecimal

BRSET0 5 3 DIR — Number of Cycles / Opcode Mnemonic / Number of Bytes/Addressing Mode

6

CISC: CISC architektúra je charakterizovaná zložitou inštrukčnou sadou. Inštrukcie trvajú viacero SC. Inštrukcie sú veľmi výkonné. Niektoré sú tak zložité, že ich využitie je minimálne. Len zaberajú na čipe miesto. Jedna z inštrukcií je aj NOP.

Prvé procesory sa vyrábali s architektúrou CISC, pomenované boli takto až keď sa vymyslela architektúra RISC.

Napr. fy Motorola vyrábala 8-bitový mikrokontroler **68HC05**.

RISC:

TABLE 9-2: PIC16CXX INSTRUCTION SET

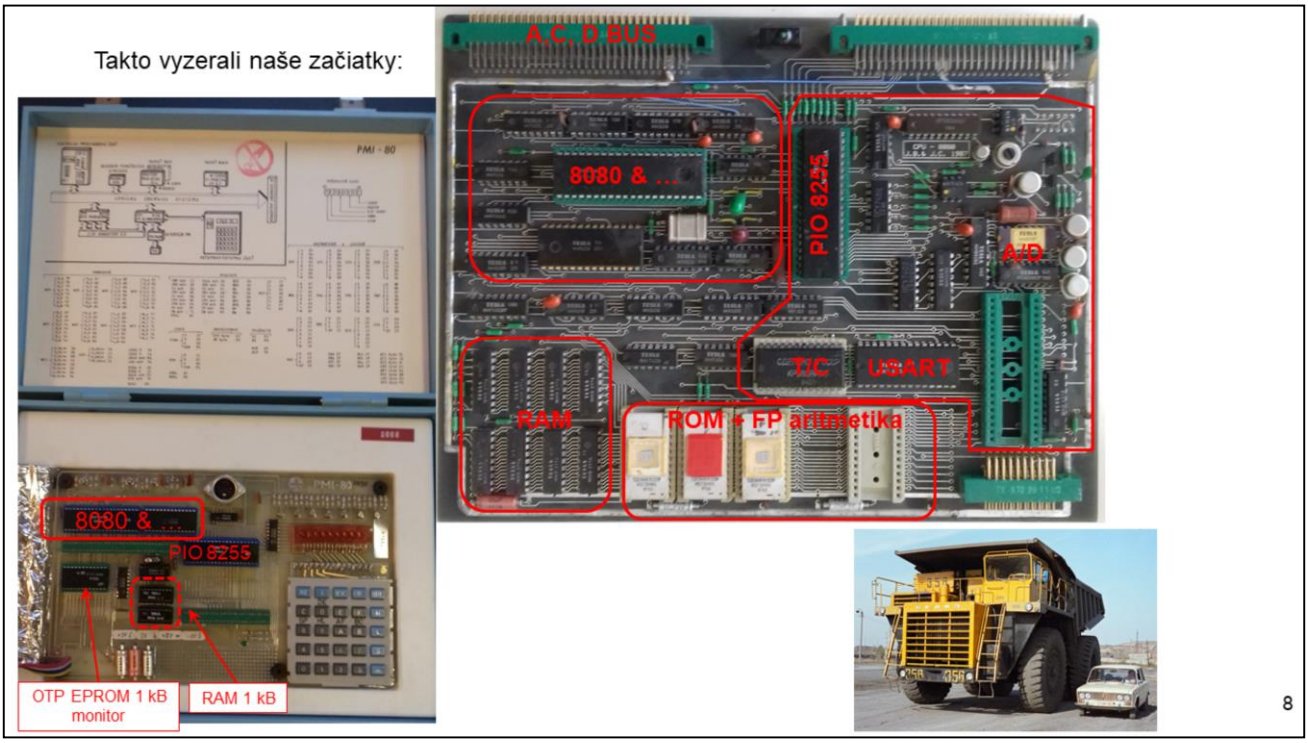
Mnemonic, Operands		Description	Cycles	14-Bit Opcode MSb	LSb	Status Affected	Notes
BYTE-ORIENTED FILE REGISTER OPERATIONS							
ADDWF	f, d	Add W and f	1	00 0111 dfff ffff		C,DC,Z	1,2
ANDWF	f, d	AND W with f	1	00 0101 dfff ffff		Z	1,2
CLRF	f	Clear f	1	00 0001 lfff ffff		Z	2
CLRW	-	Clear W	1	00 0001 0xxx xxxx		Z	
COMF	f, d	Complement f	1	00 1001 dfff ffff		Z	1,2
DECF	f, d	Decrement f	1	00 0011 dfff ffff		Z	1,2
DECFSZ	f, d	Decrement f, Skip if 0	1(2)	00 1011 dfff ffff			1,2,3
INCF	f, d	Increment f	1	00 1010 dfff ffff		Z	1,2
INCFSZ	f, d	Increment f, Skip if 0	1(2)	00 1111 dfff ffff			1,2,3
IORWF	f, d	Inclusive OR W with f	1	00 0100 dfff ffff		Z	1,2
MOVF	f, d	Move f	1	00 1000 dfff ffff		Z	1,2
MOVWF	f	Move W to f	1	00 0000 lfff ffff			
NOP	-	No Operation	1	00 0000 0xx0 0000			
RLF	f, d	Rotate Left f through Carry	1	00 1101 dfff ffff		C	1,2
RRF	f, d	Rotate Right f through Carry	1	00 1100 dfff ffff		C	1,2
SUBWF	f, d	Subtract W from f	1	00 0010 dfff ffff		C,DC,Z	1,2
SWAPF	f, d	Swap nibbles in f	1	00 1110 dfff ffff			1,2
XORWF	f, d	Exclusive OR W with f	1	00 0110 dfff ffff		Z	1,2
BIT-ORIENTED FILE REGISTER OPERATIONS							
BCF	f, b	Bit Clear f	1	01 00bb bfff ffff			1,2
BSF	f, b	Bit Set f	1	01 01bb bfff ffff			1,2
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	01 10bb bfff ffff			3
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	01 11bb bfff ffff			3
LITERAL AND CONTROL OPERATIONS							
ADDLW	k	Add literal and W	1	11 111x kkkk kkkk		C,DC,Z	
ANDLW	k	AND literal with W	1	11 1001 kkkk kkkk		Z	
CALL	k	Call subroutine	2	10 0kkk kkkk kkkk			
CLRWDT	-	Clear Watchdog Timer	1	00 0000 0110 0100		$\overline{TO}$,$\overline{PD}$	
GOTO	k	Go to address	2	10 1kkk kkkk kkkk			
IORLW	k	Inclusive OR literal with W	1	11 1000 kkkk kkkk		Z	
MOVLW	k	Move literal to W	1	11 00xx kkkk kkkk			
RETFIE	-	Return from interrupt	2	00 0000 0000 1001			
RETLW	k	Return with literal in W	2	11 01xx kkkk kkkk			
RETURN	-	Return from Subroutine	2	00 0000 0000 1000			
SLEEP	-	Go into standby mode	1	00 0000 0110 0011		$\overline{TO}$,$\overline{PD}$	
SUBLW	k	Subtract W from literal	1	11 110x kkkk kkkk		C,DC,Z	
XORLW	k	Exclusive OR literal with W	1	11 1010 kkkk kkkk		Z	

7

RISC: RISC architektúra má jednoduchú inštrukčnú sadu, ktoré trvajú jeden alebo dva SC. Počet adresovacích módov RISC inštrukcií je minimalizovaný. Výsledkom je veľmi rýchle vykonávanie inštrukcií. Počítače s architektúrou RISC majú veľmi jednoduchý dekóder inštrukcií.

Mikrokontrolery, MMP sa prevážne vyrábajú s architektúrou RISC. Aj napriek tomu, že inštrukčná sada je redukovaná, inštrukcia **NOP** nevypadla zo zoznamu. Inštrukcia **NOP** (ne)robí nič. Len generuje oneskorenie o 1 SC.

Atmel AVR aj keď sú to podľa výrobcu RISC mikrokontrolery majú zvláštnu inštrukčnú sadu. Skôr by som povedal, že je to mix. Viac si o nej povieme na cvičeniach.



V roku 1982 vyrobili v TESLE Piešťany jednoduchý 8-bitový jednodoskový výukový mikropočítač.

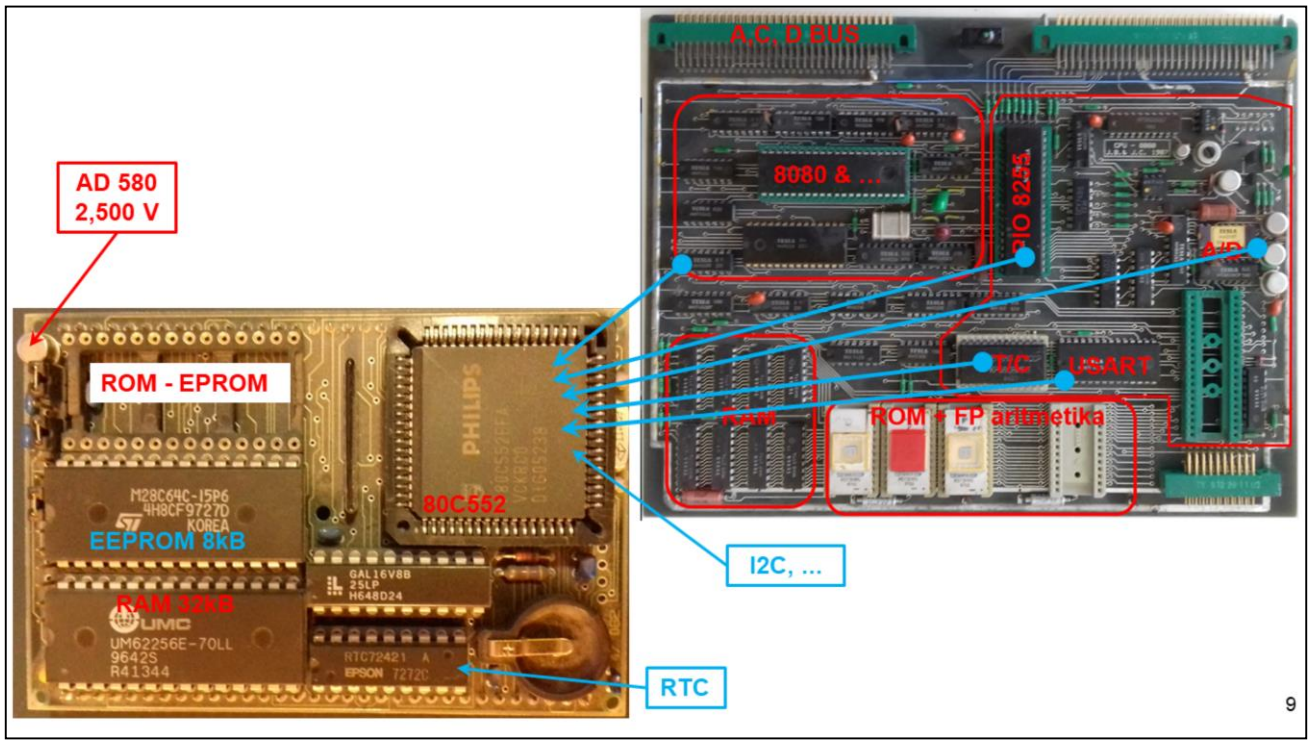
V pamäti PROM bol uložený *monitor strojového kódu,* ktorý umožňoval:

- Kláves RE. RESET systému. Obsah pamäte RAM sa nezmenil.
- Kláves R. Prezeranie/zmena obsahu registrov.
- Kláves M. Prezeranie/zmena obsahu pamäte .
- Kláves EX. Štart vloženého programu od nastavenej adresy.
- Kláves BR. Breakpoint.
- Kláves I. Obsluha externého prerušenia.
- Kláves L a S. Zápis/čítanie dát na pripojený magnetofón.

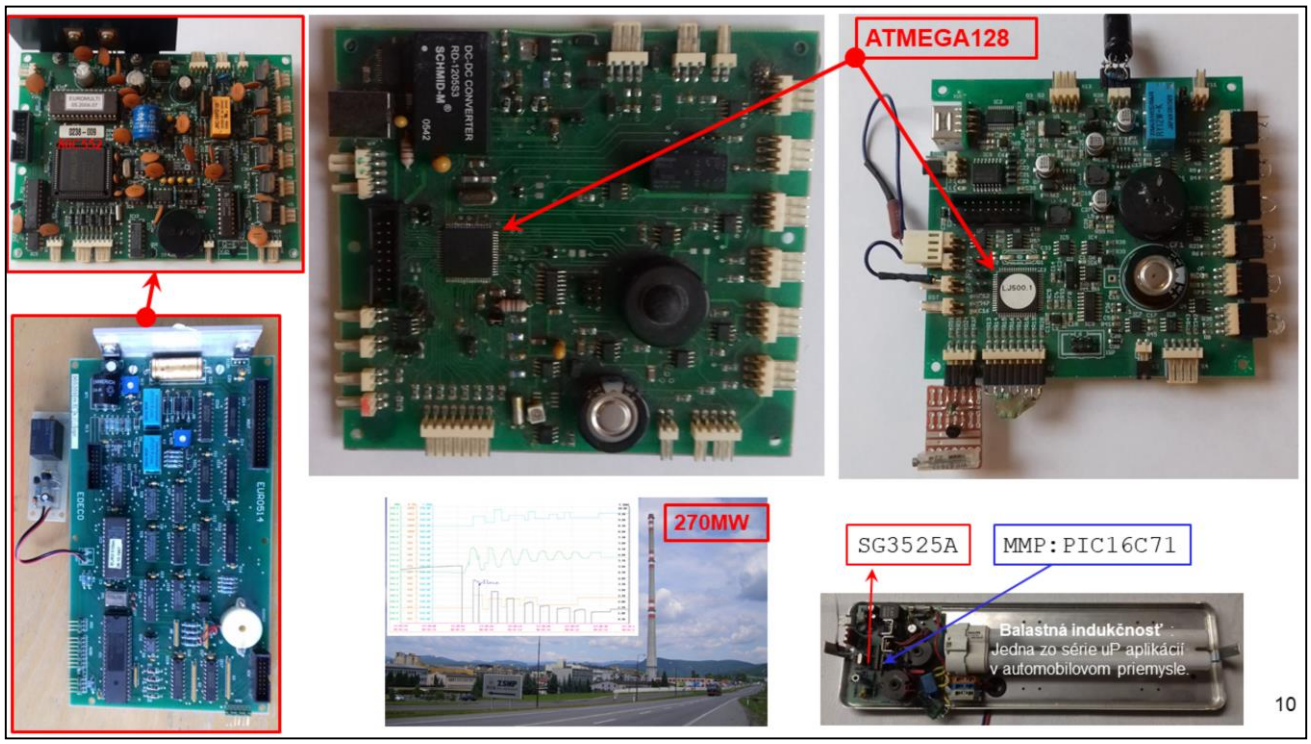
Ako zaujímavosti uvediem. Nastavenie SP bolo po pripojení napájania náhodné. Ak SP ukazoval do RAM-ky mohol program fungovať. Použitá klávesnica nebola ošetrená voči zákmitom, alebo len slabo.

Ak sme potrebovali vložiť inštrukciu, nedalo sa. Zvyšok sa musel prepísať. Teda MONITOR nemal zabudované také základné funkcie editora ako INS/DEL. A pripomeňme, že na samotné programovanie sa nepoužil assembler, ale kódy inštrukcií v HEXA kóde.

Neskôr pre nás navrhli kolegovia jednodoskový systém s aritmetikou v pohyblivej rádovej čiarke a pomocou neho sme riadili vo vozidle BELAZ prenos výkonu s dieselového motora cez generátor do MOTOR kolies. Dnes by sme to nazvali hybrid.

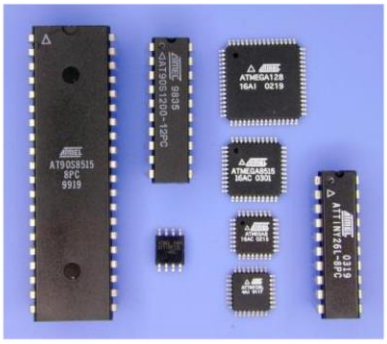


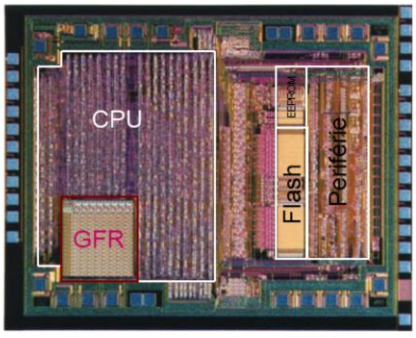
Časom sa mnohé externé súčiastky „objavili“ na čipe procesora a niekoľko rokov sme využívali náš klon dosky dole vľavo. Hustota súčiastok (plocha súčiastok/ plocha dosky) bola „viac“ ako jedna. Pod RAM a ROM-kou boli ešte ďalšie súčiastky.



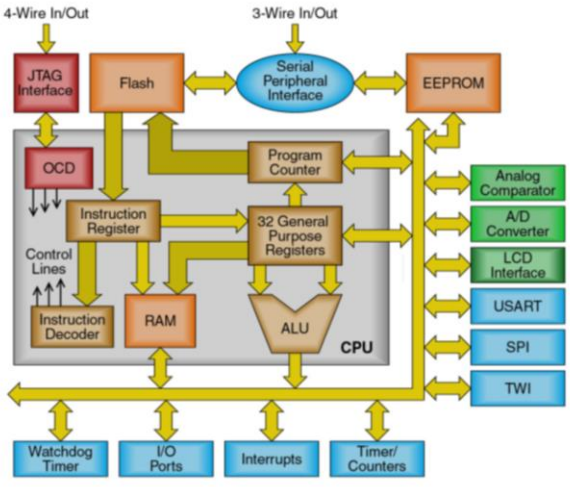
Spolupracovali sme aj s porevolučnou praxou. Riadiaci systém trezorovej pokladne postavený na báze 8035, sme prerobili najskôr s procesorom 80C552 a neskôr s AVR ATMEGA 128. Navrhovali sme aj veľmi jednoúčelové riadiace systémy, napr. tzv. balastnú indukčnosť, kde sme implementovali všetky funkcie, ktoré zadávateľ požadoval a originálny obvod SG3525A ich nemal implementované. Napr.: „tri krát a dosť“. Naviac sme dorobili funkciu stmievanie.

Veľa krát sme len pomáhali s doladením algoritmov pre konkrétne počítačové riadenie. Jeden zo spôsobov zvyšovania účinnosti zariadení je nahradiť málo efektívnu analógovú techniku počítačovým riadiacim systémom. Výsledkom bola nestabilita systému. Stačilo spojiť dve veci: „Počítač“ je sériový stroj a trochu vedomostí z TR. Ináč povedané: Trúfli sme si aj na cca jednu desatinu inštalovaného elektrického výkonu Slovenska.





AVR - Architektúra



11

Jadro procesora: CPU kontrolera sa skladá z: arithmeticko logickej jednotky (aritmetické, logické, bitové operácie, ...), riadiacej jednotky, a registrov (stack pointer - SP (16b), program counter - PC (pre ATMEGA328 (14b)), špeciálne registre, . . .).

Pamäť: je rozdelená na pamäť programu (FLASH EEPROM) a pamäť dát (statická RAM).

Digital I/O: Základ vlastností vstupno/výstupných pinov (Input/Output) = (I/O) je odvodený od obvodu 8255. Tento obvod je tvorený 3-mi 8-bitovými I/O portami. Sú označené ako PA, PB, a PC. Tieto porty sa dajú „rôzne“ konfigurovať a môžu pracovať v rôznych módoch. Obvod 8255 je TTL kompatibilný.

Radič prerušení: Nie je súčasťou von Neumanovej koncepcie počítača. Nie je to nutná časť jeho architektúry. Treba povedať, že prerušovací podsystém podstatne urýchľuje výpočty.

Timer/Counter: Možno ich nakonfigurovať ako počítadlá udalostí (interné/externé). Možno ich použiť ako časovače, ako generátory PWM

signálov. Základom všetkých T/C (P/C) je obvod 8253. Obsahuje 3 16-bitové nezávislé počítadlá s predvoľbou. Ak počítadlo načítava ext. impulzy hovoríme o ČITAČI. Ak je na vstup počítadla pripojený presný zdroj frekvencie hovoríme o ČASOVAČI. P/C majú od svojho vzniku dva nedostatky.

1. „Pretečenie“ a
2. V 8-bitových MMP 16-b hodnotu prednastavujeme/vyčítavame cez 8-bitovú dátovú zbernicu. Treba použiť tzv. pomocný záchytný register. Problém, je že ho súčasne môžu použiť dve periférie. Takáto chyba sa nedá odladiť. Pretože musíte súčasne „spustiť“ dve udalosti vo veľmi krátkom čase. Až potom sa prejaví.

Analog I/O: Väčšina mikrokontrolerov má zabudovaný 8, 10, ... bitový A/D prevodník s postupnou aproximáciou pred ktorým je zapojený analógový multiplexer. Ako D/A prevodník sa v jednoduchších mikrokontroleroch používa PWM výstup.

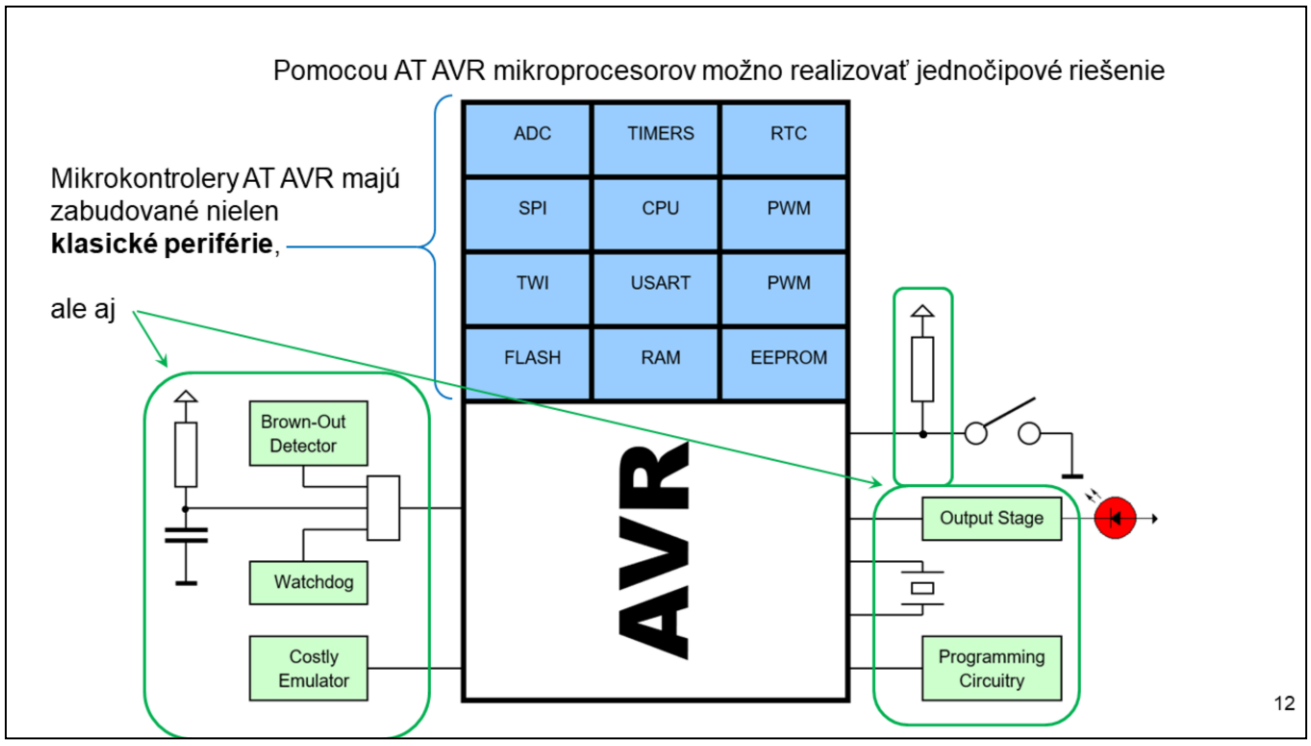
Komunikačné zbernice Interface: mikropočítače majú aspoň jeden sériový interface pomocou ktorého možno komunikovať s PC.

Na komunikáciu s inými periférnymi obvodmi sa používa napr. SPI, I2C, resp. CAN zbernica.

Watchdog Timer: Pôvodne bol resetovací obvod externou súčiastkou. Jej súčasťou bol aj tzv. watchdog timer. Jeho úlohou je zabezpečiť trvalý beh programu. Riadiaci systém reálneho času má technológom stanovený najdlhší čas, ktorý je vyhradený na vykonanie jednotlivých častí programu. WDT je na začiatku nejakej časti programu prednastavený, a ak do vypršania času nie je v ďalšej časti programu prednastavený, generuje reset systému vyvolaný WDT-om. A celý systém sa uvedie do „bezpečného“ stavu.

Debugging Unit: Táto jednotka nám umožňuje viac menej v reálnom čase sledovať beh programu. Jednoducho povedané je to hardwarový simulátor.

...



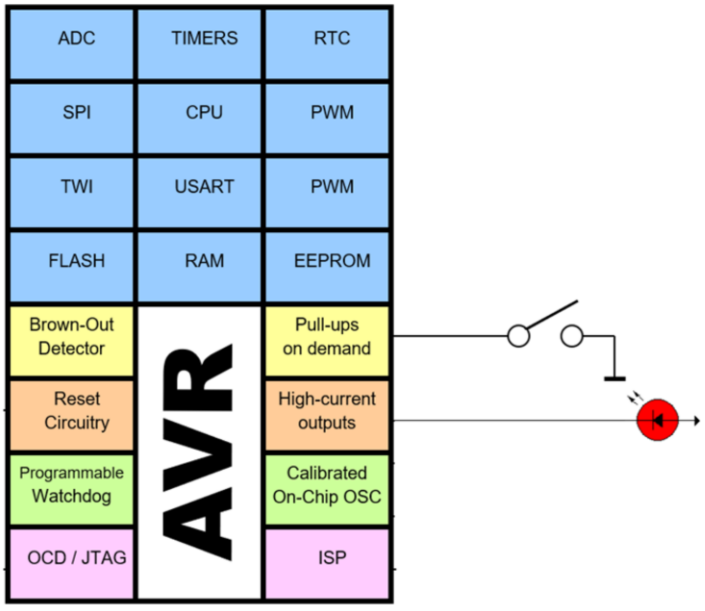
Na tomto prevzatom obrázku je ukázané, čo všetko je zabudované do jedného celku.

Dokonca aj programátor. Ladenie programu bola operácia pozostávajúca s nasledovných krokov.

- Preklad programu. Vytvorenie HEX súboru.
- Naprogramovanie pamäte EPROM. Zasunutie pamäte a odskúšanie.
- Ak sme neboli z výsledkom spokojný. Nasledovalo vymazanie obsahu EPROM pamäte UV svetlom (niekoľko minút).
- A mohli sme pokračovať.

Dnes tento cyklus trvá: **A je to**. Potrebujete len jeden káblik.

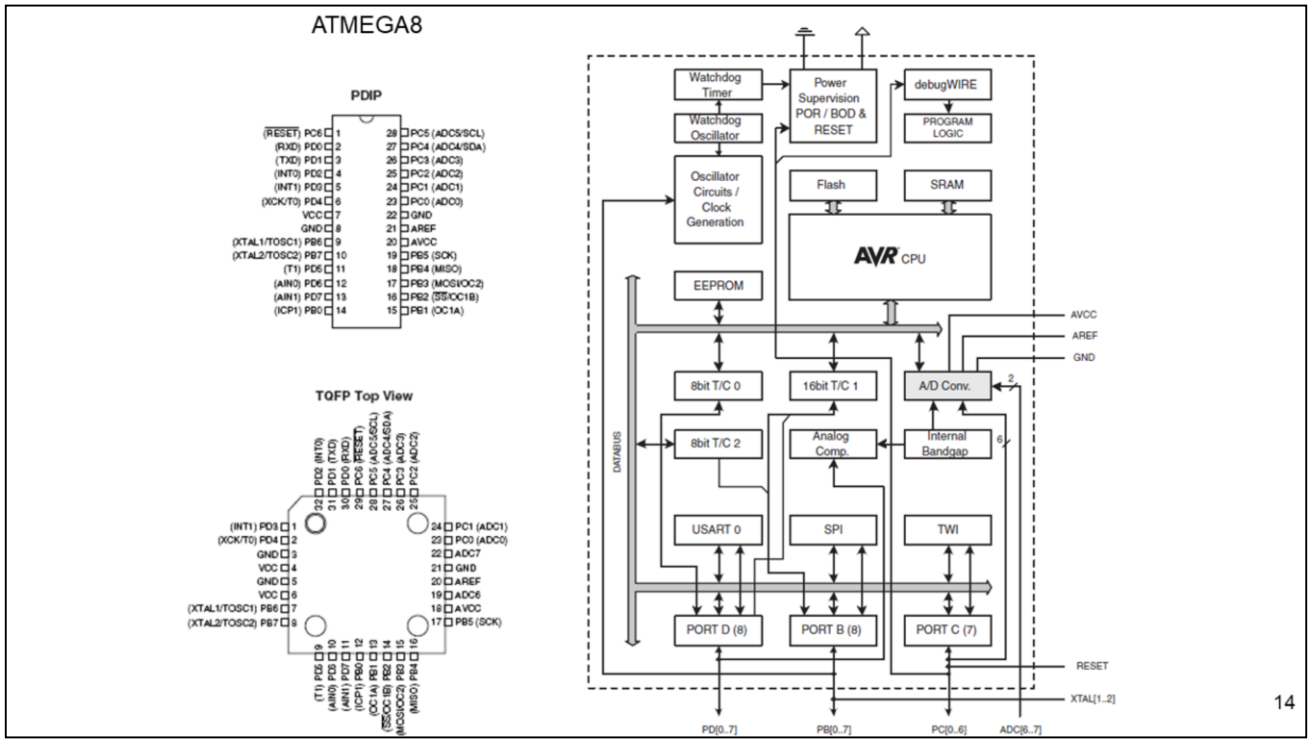
Pomocou AT AVR mikroprocesorov možno realizovať jednočipové riešenie



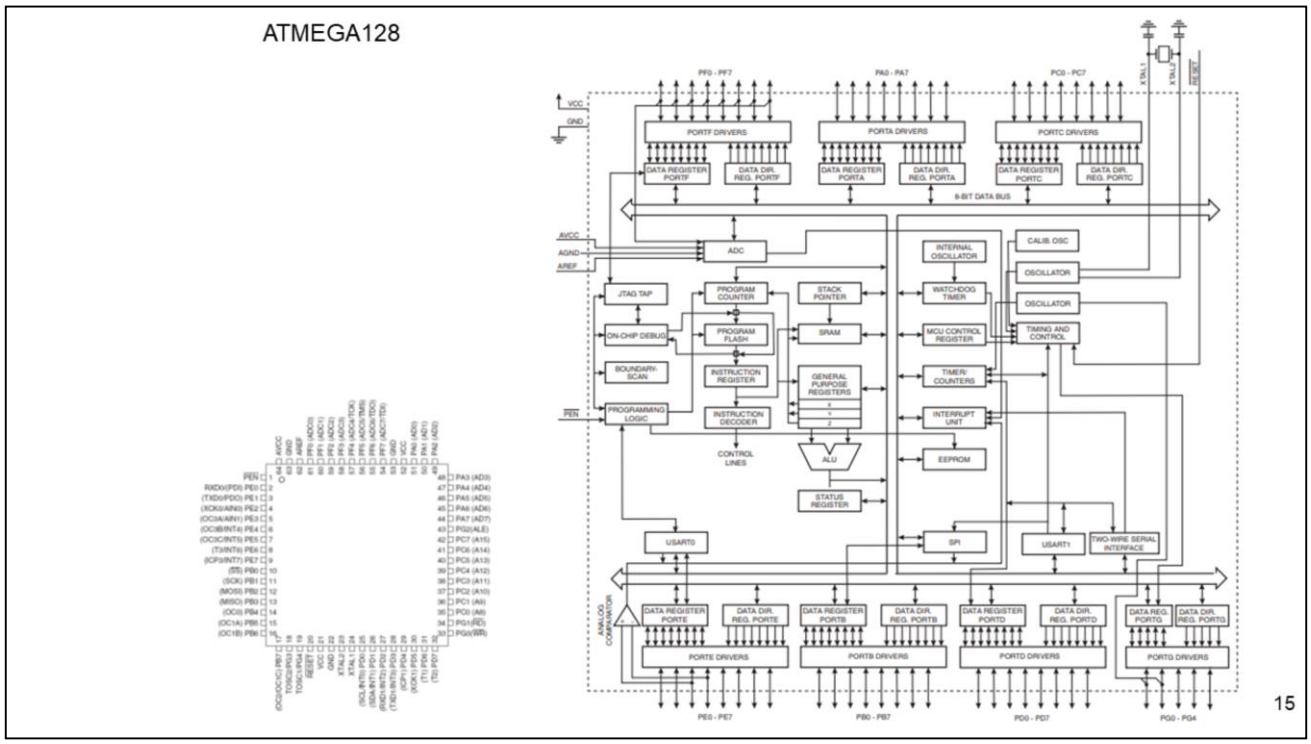
13

Mikrokontroler použitý na cvičeniach a prednáškach má okrem bežných periférií, už sme spomenuli,

aj obvody na sledovanie poklesu napájania, resetovacie obvody a watchdog, obvody umožňujúce testovať stav periférií za behu programu, súčasťou je aj kalibrovaný oscilátor, niektoré mikrokontrolery majú zabudovaný aj zdroj referenčného napätia pre A/D prevodník, a možno som aj na niečo zabudol.



Na obr. je uvedená bloková štruktúra 8-bitového procesora ATMEGA8.

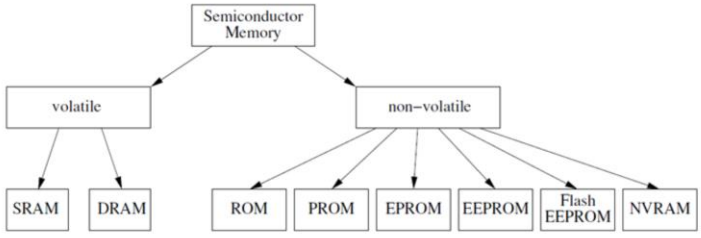


O niečo zložitejšia bloková schéma odpovedá ATMEGA 128.

Jednotlivé typy AVR mikrokontrolerov sa od seba líšia hlavne počtom I/O portov, počtom zdrojov prerušenia, počtom zabudovaných T/C a ich možnosťami využitia. Veľkosťou pamätí: programu - FLASH, dát RAM a EEPROM.

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 !!!

16

Dnes sa ako pamäte počítača výhradne používajú polovodičové pamäte. Podrobne sme si o nich hovorili na predmete základy počítačov.

Pamäte typu volatile – prchavé, stratia svoj obsah po odpojení napájania.

Potom je namieste otázka prečo nepoužívame len non-volatile pamäte? Veď informáciu udržia a vieme z nich aj čítať aj do nich zapisovať.

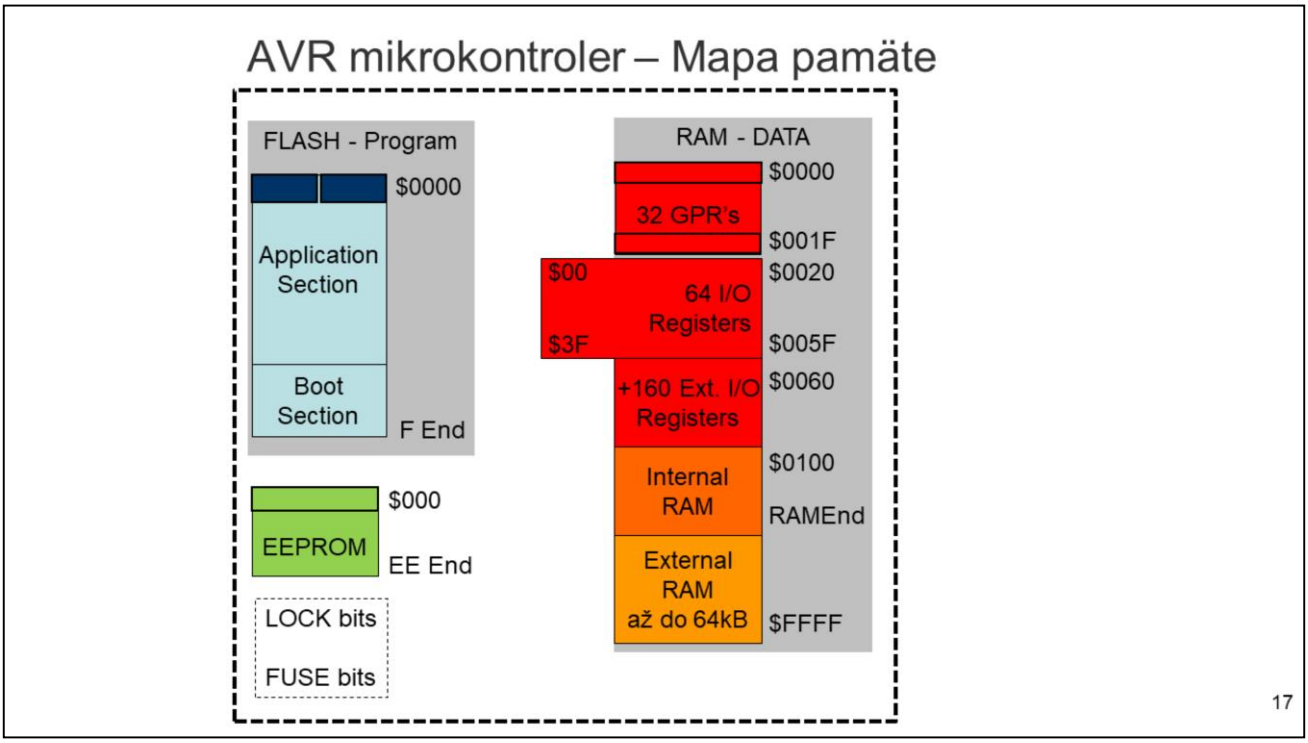
Problém je so zápisom a následným čítaním. Išlo by to, ak by sme vedeli na chvíľu pozastaviť taktovanie zberníc.

Na začiatku (2. polovica 20 storočia) sa vyrábali ako veľkokapacitné pamäte napríklad páskové magnetické pamäte. U nich sme sa k informácii dostali, iba vtedy, keď sme si prečítali predchádzajúcu informáciu (niečo ako posuvný register). Tieto pamäte sa nazývali sekvenčné.

Ich protipólom boli pamäte kde sme sa k informácii dostali viac menej náhodne mohli sme prečítať a zapísať ľubovoľné pamäťové miesto. Tieto pamäte dostali názov RWM-RAM (Read-Write Memory - Random Access Memory). Časom sa skratka pamäte skrátila na RAM. Slovo **náhoda** neznamená, že do prístupu k pamäti je zahrnutá náhoda.

V pamäti RAM sa väčšinou ukladajú dáta. Pamäťové miesta sú realizované ako preklápacie obvody, ktoré po výpadku napájania informáciu stratia. Program sa ukladá do pamätí, ktoré svoj obsah uchovávajú aj po odpojení napájania. A nazývajú sa ROM. Dnes sa v mikrokontroleroch používajú ako pamäť programu pamäte EPROM a Flash EEPROM. V MMP sa pamäte RAM vyrábajú ako statické pamäte RAM – SRAM.

Blízko CPU je niekoľko špeciálnych pamäťových miest. označované sú ako registre a majú špeciálne určenie.



Na obrázku sú nakreslené blokovo základné pamäte AVR mikrokontrolerov.

Pamäť programu je organizovaná po slovách a fyzicky je rozdelená na dve základné časti. Aplikačná a tzv. bootovacia.

Pripomeňme si: S pamäti EEPROM môžeme spustiť vykonávanie programu, ale keď do nej zapíšeme, nedá sa z nej chvíľu informácia čítať. To by spôsobilo kolaps programu. To je dôvod fyzického rozdelenia pamäte programu na dve časti.

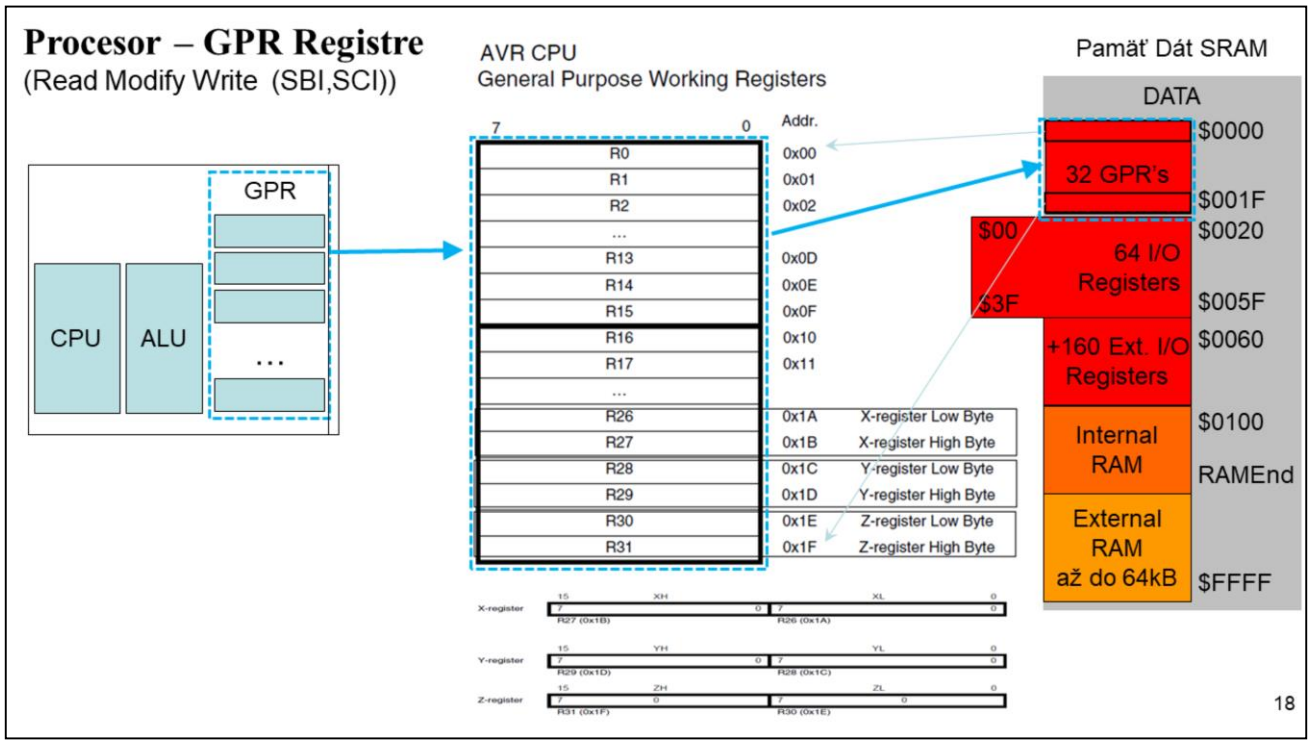
Pamäť dát – RAM-ka je bytovo organizovaná. Trocha predbehneme. Každá zabudovaná periféria má vstupné a výstupne registre, riadiace registre a stavové registre. Tie sú mapované aj do pamäte dát – RAM-ky. Preto ten komplikovaný obrázok. RAM-ka slúži na dočasné odkladanie spracovávaných informácií.

Pamäť EEPROM. Obsah tejto pamäte zostane zachovaný aj po výpadku napájania. Z tohto dôvodu sa do nej ukladajú informácie trvalého charakteru, ale ktoré treba čas od času zmeniť. Napríklad konštanty P, I a D regulátora.

Základné parametre pamätí EEPROM:

- kapacita 64B až 8kB
- netreba externé komponenty,
- je lokalizovaná mimo adresný priestor programu a dát
- zápis cca 8ms (počet erase/write cyklov 100 000 (1M)), súčasťou je aj mazanie; okamžité čítanie
- zapojená ako periféria s GPR registrami

Špeciálna EEPROM: LOCK bits a FUSE bits. Pripomeňme si. Oscilátor môže biť kryštálový - externý, resp. interný kalibrovaný RC oscilátor. Táto konfigurácie sa nastavuje v týchto bitoch. Inou takou možnosťou je zabránenie nechcenému vyčítaniu obsahu pamäte programu, resp. EEPROM. Môžu v nej biť uložené prístupové kódy k citlivým informáciám. Aj toto sa dá nastaviť. Ak by bol watchdog spustený automaticky po uvoľnení RESET-u, asi by sme nikdy program neodladili. Čo všetko sa dá takto nastaviť, viď KL daného mikropočítača.



Registre (8-b údaj): špeciálne pamäťové miesta, ktoré sú:

- priamo spojené s CPU,
- možno ich priamo použiť v asemblerovských inštrukciách, buď ako cieľový register alebo ako register z ktorého čítame alebo môže slúžiť na prenos informácie,
- na operácie s ich obsahom sa používajú jednoslovné inštrukcie.

AVR má 32 registrov a majú označenie R0 až R31. Možno ich pomocou asemblerovskej directivy premenovať. Špeciálnu funkciu majú registrové páry R26:R27, R28:R29 a R30:R31. Pomenované sú: X, Y a Z. Sú to 16-bitové pointer registre. Možno pomocou nich adresovať 65536 pamäťových miest v SRAM alebo v pamäti programu.
Podstatná výhoda RISC procesorov. Všetky aritmetické a logické operácie sa vykonávajú v GPR „General Purpose Registers“ (registre pre obecné použitie). GPR a I/O (ďalej) 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.

Pr.1. :

```
.DEF MojRegister = R16
        LDI  MojRegister,110
```

Pr. 2. :

```
.DEF MR1 = R16
.DEF MR2 = R15
        LDI  MR1,110
        MOV  MR2, MR1          ;  Prvý register je cieľový, MR2 = MR1
```

Ak by sme program zapísali takto

Napr.3 :

```
.DEF MR1 = R16
.DEF MR2 = R15
        LDI   MR2,110
        MOV  MR1, MR2
```

Tento program by nefungoval správne, pretože len do registrov R16 až R31 možno dáta vložiť priamo pomocou LDI inštrukcie.

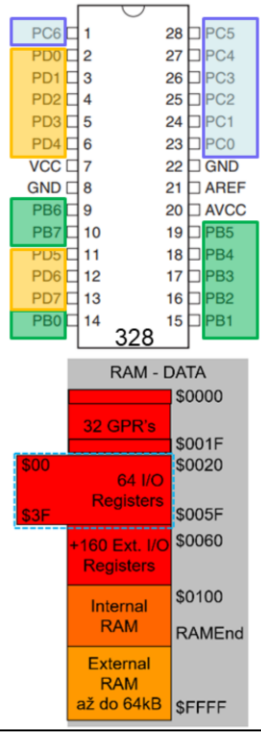
Toto pravidlo má výnimku: CLR MR2 „funguje“. Podobnú výnimku ako LDI majú aj ďalšie inštrukcie.

Zapíšte si do pamäti: V prvých rokoch práce s MMP je na 101% chyba vomne. A ten zvyšok do 100% je chyba na strane MMP. Neskôr tie čísla môžete upraviť.

Ak budete riešiť takéto problémy *„Do not use read-modify-write instructions (*SBI *and* CBI*) to set or clear the* MPCM *bit. The* MPCM *bit shares the same* I/O *location as the* TXC *flag and* ***this might accidentally be*** *cleared when using* SBI *or* CBI *instructions.“*, ste dobrý.

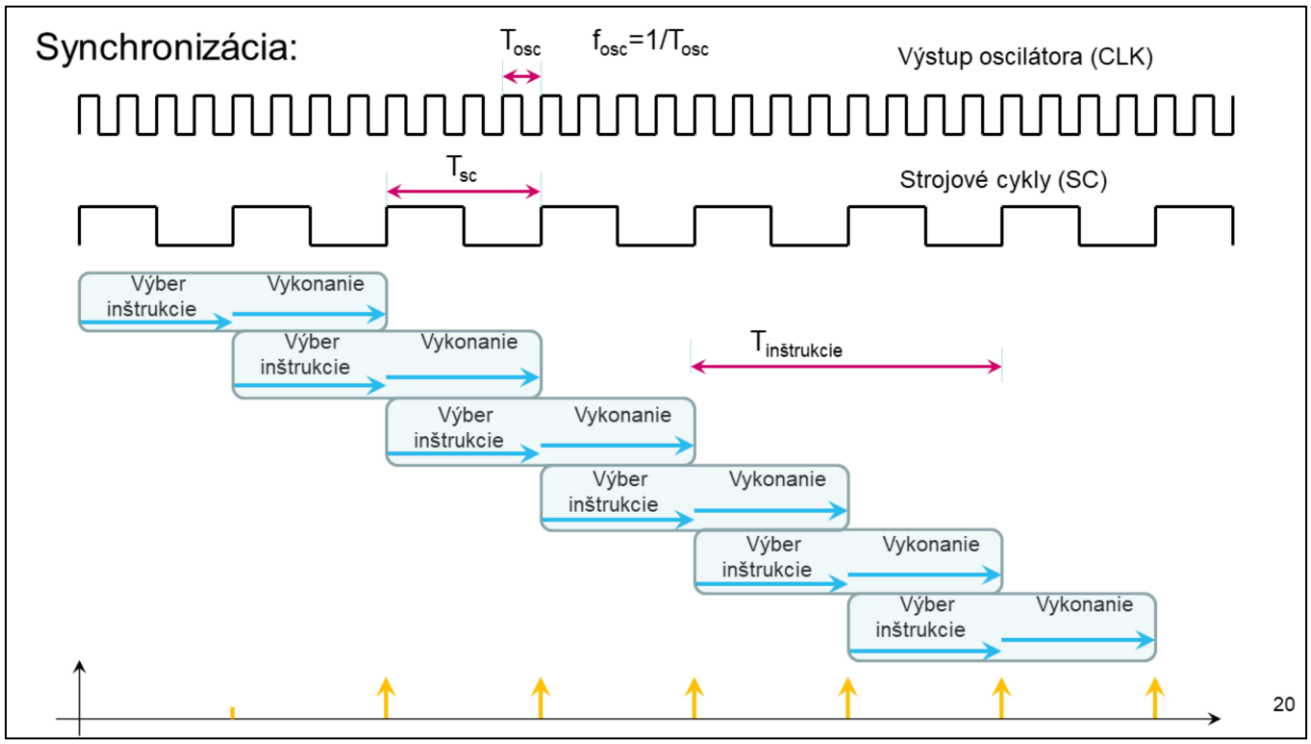
I/O Registre – porty (Read Modify Write (SBI,SCI))

- **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 (0b1111 0000), to znamená bity 7- 4 v PORTA sú výstupné a bity 3 - 0 sú 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,
    - vstupy a výstupy zberníc: USART, I2C, ...



19

Namiesto písmenka x dosadzujeme písmeno konkrétneho portu. Teda, A, B, C, ...



Počítač ako taký môžeme charakterizovať ako synchronný sekvenčný obvod. Učili sme sa na Základy Počítačov.

Sekvenčný obvod je taký, u ktorého výstup závisí nielen od vstupov ale aj od predchádzajúceho stavu. Sekvenčné obvody rozdeľujeme na asynchronné (ASO) a synchronné (SSO): Ak sa zmena výstupu vykoná okamžite po zmene vstupov, hovoríme o ASO. Ak sa zmena výstupu vykoná až s príchodom synchronizačného signálu (hodinový signál, hodiny) hovoríme o SSO.

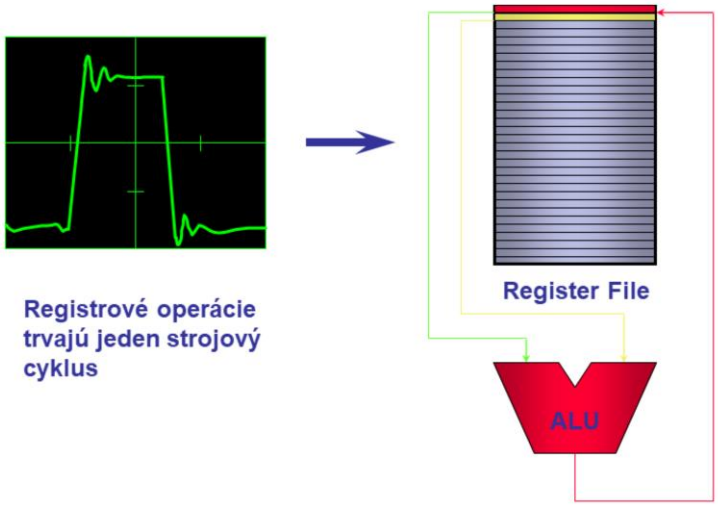
Pri SSO musíme rozlišovať či sa zmena výstupu vykoná pri zmene úrovne (zmena výstupu sa mení počas „aktívnej úrovne“) alebo pri zmene hrany (nábežná/dobežná).

Počítače potrebujú teda generátor hodinových impulzov (oscilátor). Vo všeobecnosti treba rozlišovať medzi hodinovým a strojovým cyklom.

Viď. ilustračný obrázok.

V mikrokontroleroch AVR je Tosc rovné Tsc.

Registre sú k ALU pripojené priamo



21

Časovanie inštrukcií:

Čas požadovaný k vykonaniu jednej inštrukcie je rovný hodinovému signálu procesora – trvanie SC.

Ak oscilátor procesora beží na frekvencii fosc = 4MHz potom jedna inštrukcia trvá tosc = 1/4 µs alebo 250 ns.

Pre fosc = 10 MHz je tosc = 100 ns. Tento čas je daný interným alebo externým oscilátorom.

Pripomeňme, že niektoré inštrukcie vyžadujú dva alebo viac SC. Napr. inštrukcie vetvenia.

Pr.: generovanie oneskorenia.

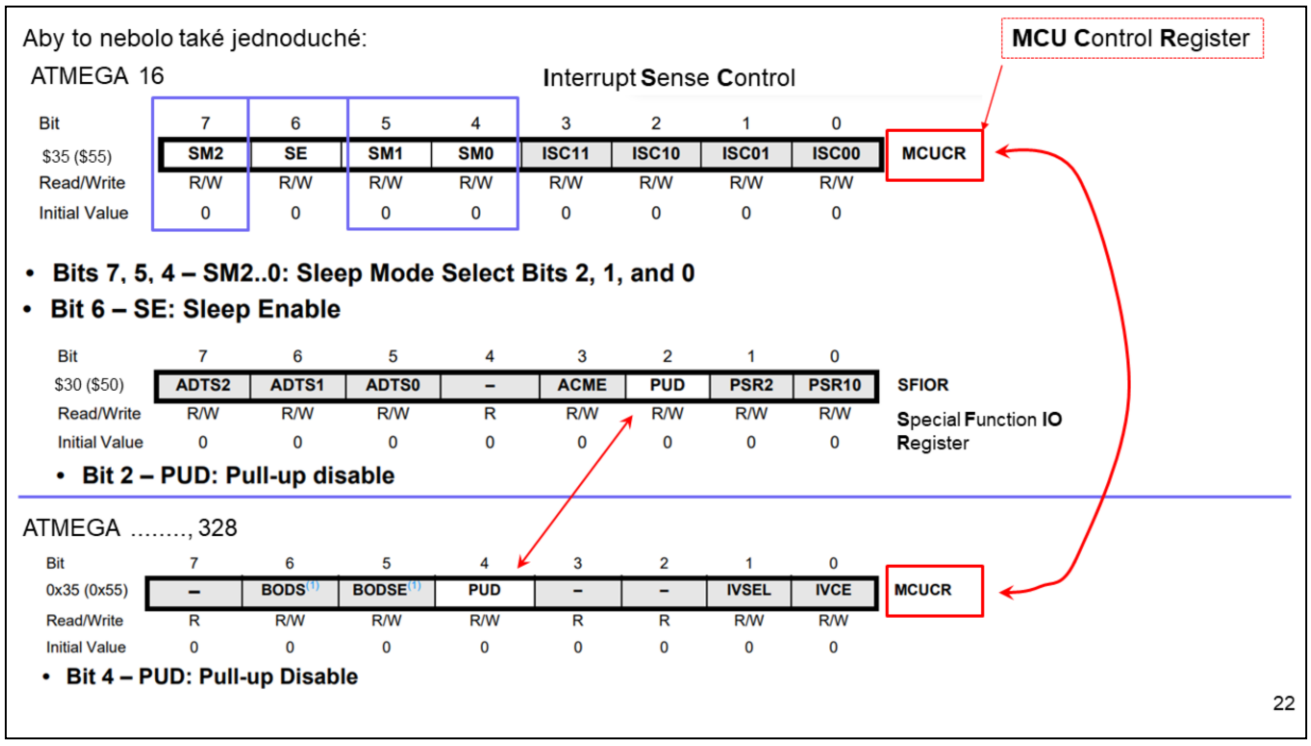
delay(x)

Niektorý 8-bit-register použijeme ako počítadlo (budeme dekrementovať pomocou instrukcie DEC):

```
        CLR R1      ; 1SC
Count:              ; „adresa“
        DEC R1      ; 1SC
        BRNE Count  ; 1SC pre goto Count, 2SC pre pokračuj,
```

Čas trvania v SC: delay = (1) + (255*2) + (1*3) = 514 SC alebo 128.5 µs pre 4 MHz.

Takže máme návod ako prednastaviť reg. R1 aby sme dosiahli napr. oneskorenie 100us.



KL niekedy dodržujú dohodu. Adresa: $aa. Data 0xdd. Niekedy aj pre adresu aj dáta použijú rovnaký spôsob zápisu HEXA hodnoty.

Porty AVR obvodov sú brány pre CPU pomocou ktorých komunikuje s internými a externými hardwerovými a softwarovými komponentami. CPU komunikuje s týmito komponentami tak, že z nich číta, resp. do nich zapisuje. Sú to vlastne I/O registre, príznakové registre a riadiace registre pôvodne periférnych obvodov. Už sme ich spomenuli.

Aj keď procesory „mladšie 80C552“ mali 128 takýchto registrov, autori AVR procesorov ich vytvorili len 64. A sú priradené k jednotlivým perifériám. Ktoré registre sú použité pre ktorý typ procesora je dané v katalógovom liste. Adresy portov konkrétnych periférií sú dané napevno. Treba povedať, že napevno pre daný typ mikrokontrolera. Napr. pre AVR 328 je PORTB na adrese 0x05 ale pre AVR MEGA128 je na adrese 0x18.

Atmega majú viac ako 64 portov. K ďalším je prístup zložitejší – iný.

Read-Modify-Write prístup k portom

Je možné nastaviť/nulovať (Set/ReSet) konkrétny bit portu bez toho aby sme ovplyvnili ostatné bity portu. K tomu slúžiaa dve inštrukcie SBI (Set Bit I/O) aCBI (Clear Bit I/O).

Vykonanie je nasledovné:

```
.EQU ActiveBit = 0                ; bit, ktorého obsah ideme meniť
        SBI PortB, ActiveBit      ; bit “ActiveBit” nastavíme do will be set to one
        CBI PortB, Activebit      ; The bit “ActiveBit” will be cleared to zero
```

Prístup k pamäťovo mapovaným portom:

Procesory s väčším počtom periférií, ako napr. ATMEGA, majú niektoré periférie mapované do priestoru pamäte RAM.

Dokonca aj k portom, ktoré sú mapované do I/O priestoru, možno pristupovať pomocou inštrukcií (ST a LD) pomocou ktorých zapisujeme/čítame zo SRAM. LEN treba pripočítať k adrese portu 0x20.

(Pripomeňme: prvých 32 addresses je rezervovaných pre GPR registre!).

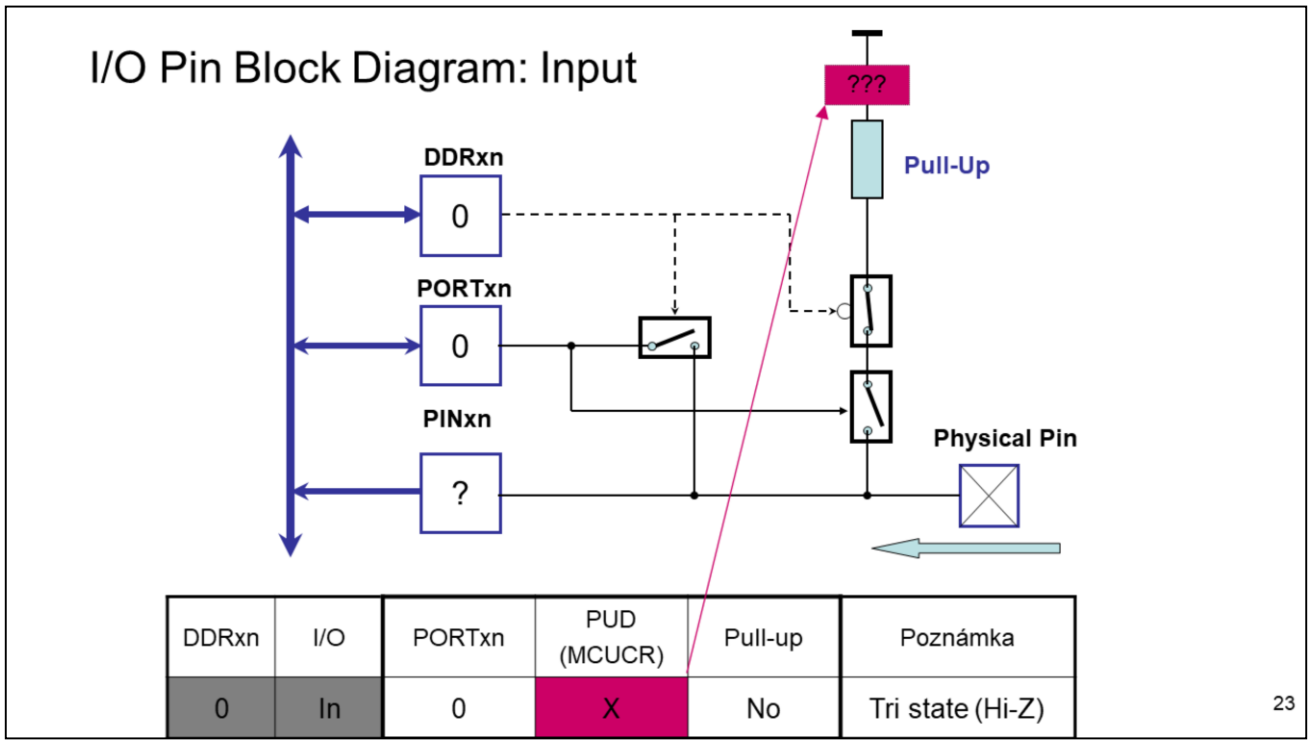
Príklad:

```
.DEF MyPreferredRegister = R16
        LDI ZH,HIGH(PORTB+32)
        LDI ZL,LOW(PORTB+32)
        LD MyPreferredRegister,Z
```

Opäť pripomeňme:

Takýto prístup je možný ale trvá dlhšie, na vykonanie treba viac inštrukcií.

To je dôvod prečo prvá adresa SRAM začína na adrese 0x60 alebo 0x100 pri väčších AVR.



Vstupné/výstupné porty

sú najjednoduchšie vstupno/výstupné rozhranie mikropočítača. Ich štruktúra a funkcia je odvodená od obvodu 8255, ktorý mal 3 8-bitové porty PORTA, B, C, ktoré sa dali naprogramovať ako vstupné, resp. výstupné a mohli pracovať v rôznych módoch.

I/O porty mikrokontrolerov (MCU, MMP) sú oveľa univerzálnejšie. Dokonca I/O informácia na pine portu môže biť binárna/analógová.

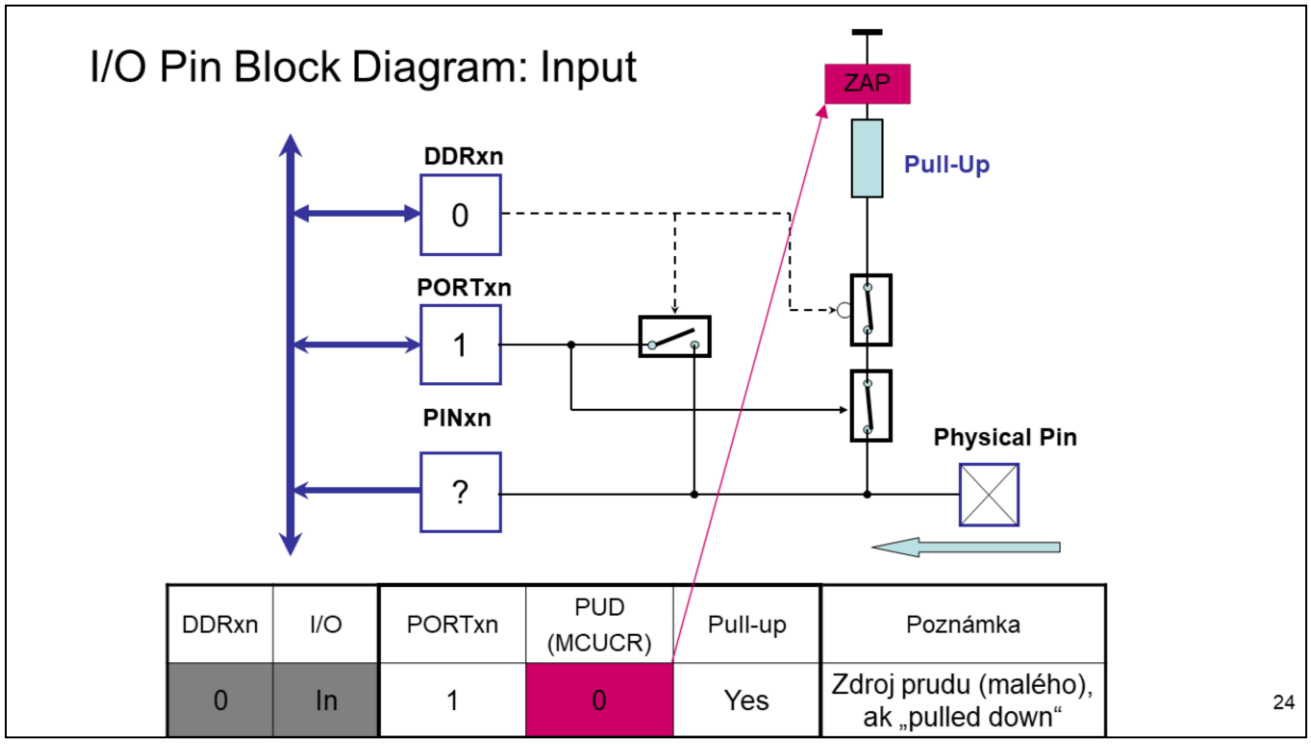
V digitálnych systémoch a teda aj v mikrokontrolleroch sa stretneme s pojmom trojstavová logika:

1. logic 0. Vývod je pripojený na 0V. Stav nízkej úrovne. (L).

2. logic 1. Vývod je pripojený na 5V. Stav vysokej úrovne.(H).

3. A ak je nejaký vývod elektricky nepripojený znamená to že je v stave vysokej impedancie (Z) alebo v tzv. treťom stave. Anglicky písaná literatúra okrem písmena Z, používa označenie: **three-state**, **tri-state**, alebo **3-state** .

Nastaveniu: DDRxn = L, PORTxn = L, a PUD = Z odpovedá konfigurácia pinu: Input, Z.



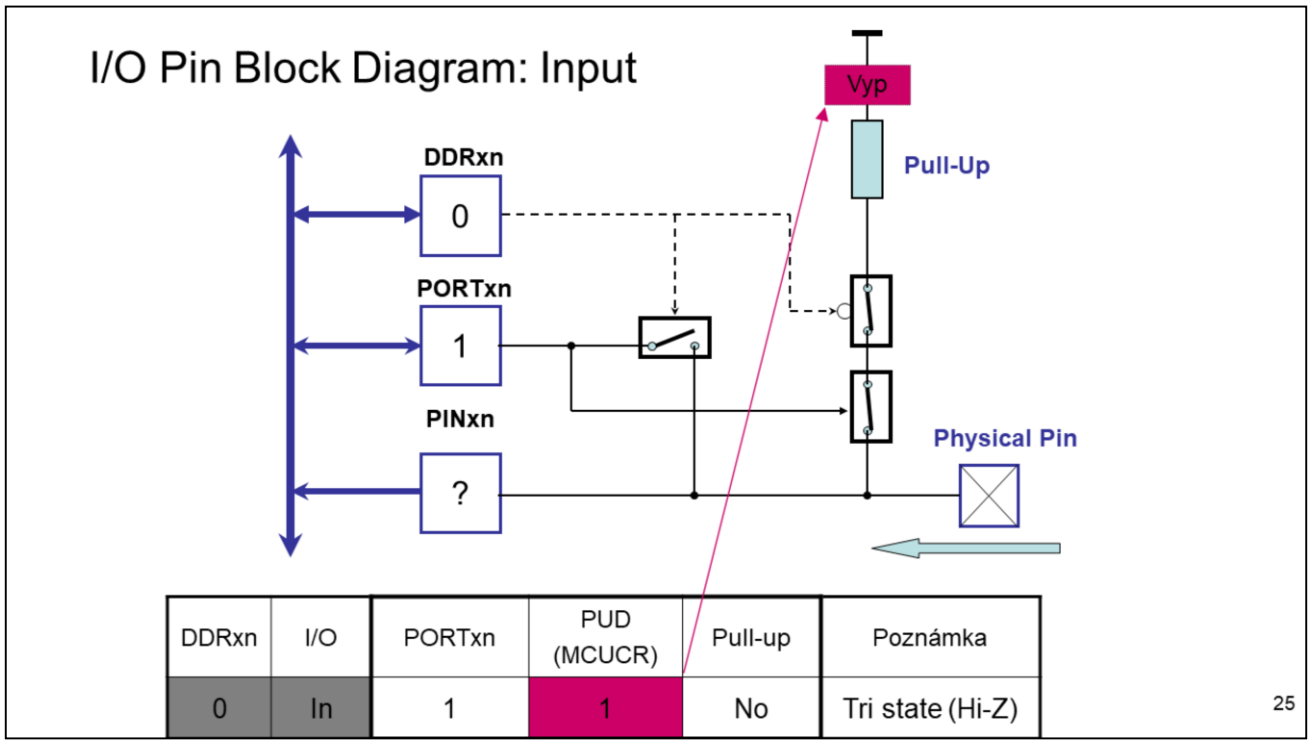
Vývoj vraj ide po špirále.

Koľko rečí vieš, toľko krát si človekom.

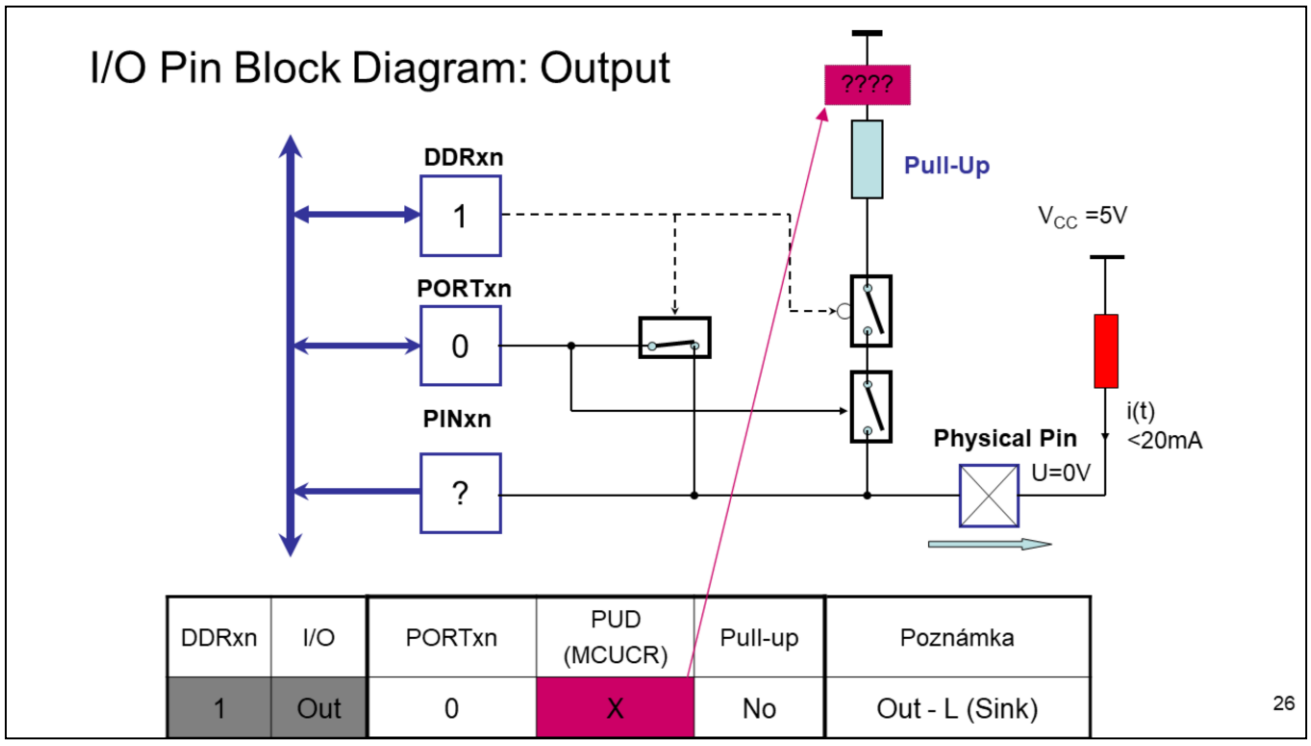
Technik by sa mal vyjadrovať na základe pravidiel – noriem. Čo si myslíte čo predstavuje tá pomlčka hore nad červeným obdĺžnikom?

Tento krát je to plus 5V. Inokedy je to GND.

Nastaveniu: DDRxn = L, PORTxn = H, a PUD = L odpovedá konfigurácia pinu: Input.

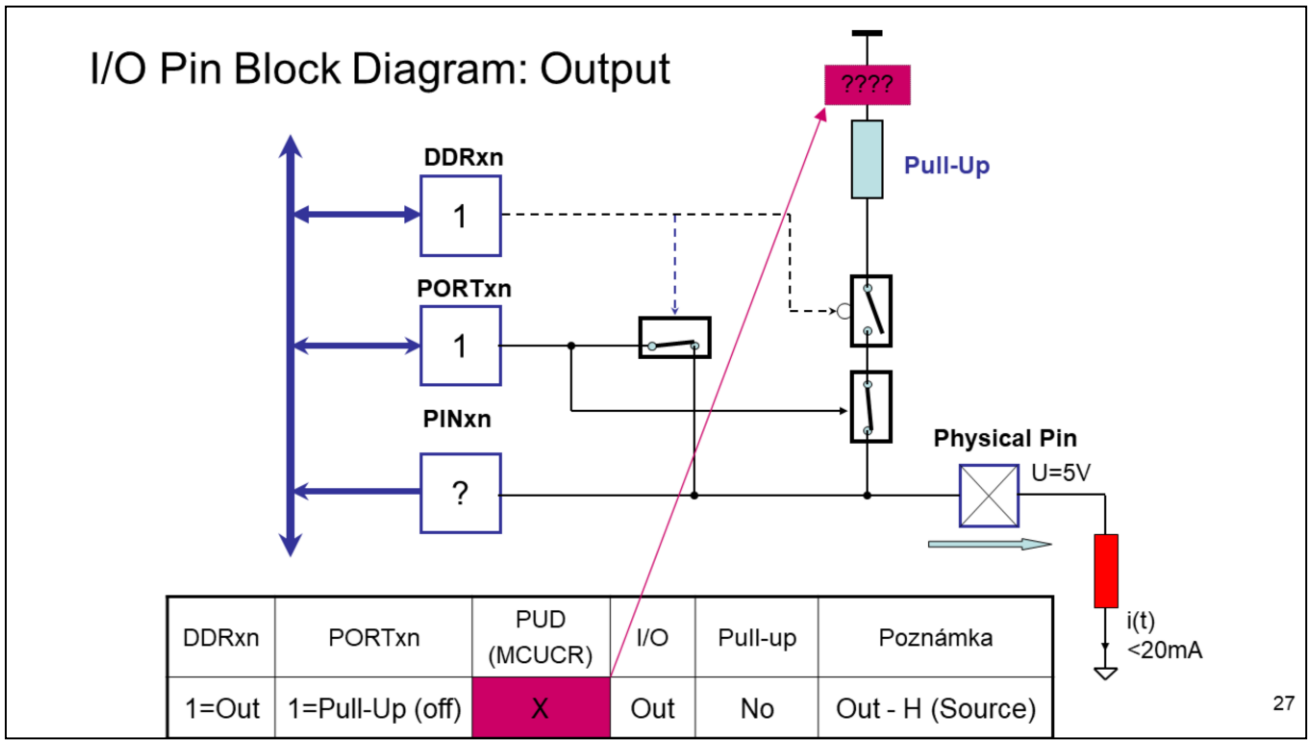


Nastaveniu: DDRxn = L, PORTxn = H, a PUD = H odpovedá konfigurácia pinu: Input, Z.



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

Nastaveniu: DDRxn = H, PORTxn = L, a PUD = ? odpovedá konfigurácia pinu: Output, L.

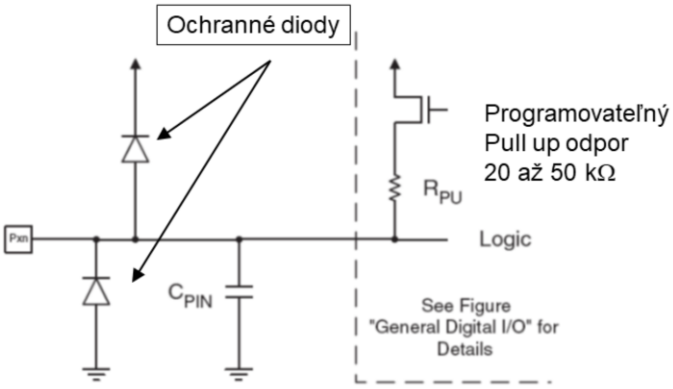


DDRxn	PORTxn	PUD (MCUCR)	I/O	Pull-up	Poznámka
1=Out	1=Pull-Up (off)	X	Out	No	Out - H (Source)

Nastaveniu: DDRxn = H, PORTxn = H, a PUD = ? odpovedá konfigurácia pinu: Output, H.

„ATMEL AVR“

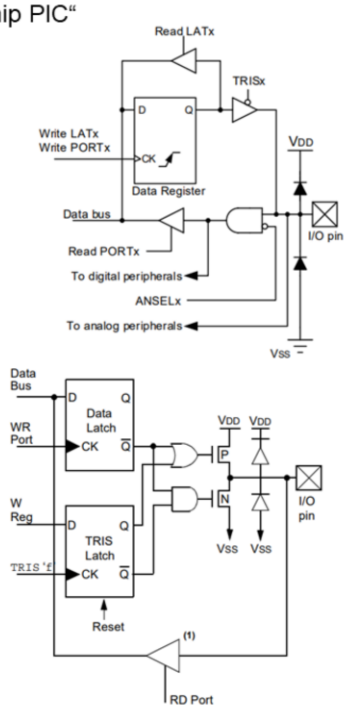
I/O Port – zapojenie vstupného pinu



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

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

„Microchip PIC“



28

Obrázok vľavo: Tu je plus (5V) tá šipka smerujúca hore.

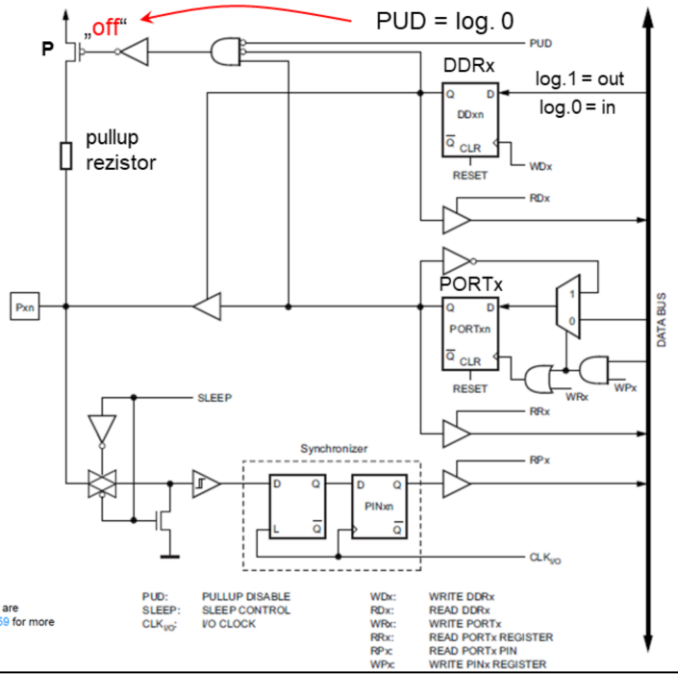
I/O Port – novšia architektúra portu:

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.

Ak niektoré I/O piny nepoužijeme, odporúča sa, aby mali definovanú úroveň. Dôvod: Zníženie spotreby.

Jeden z možných spôsobov je použiť vnútorný pullup.

Pullup počas RESETu nefunguje. Ak máme aj napriek tomu zabezpečiť minimalizáciu spotreby je vhodné použiť externý pullup resp. pulldown. Priame pripojenie k VCC alebo GND sa neodporúča, pretože to môže spôsobiť nadmerné prúdy, ak je pin náhodne nakonfigurovaný ako výstup.

Programovanie I/O bit-ov:

0x20 = 0b0010 0000

XOR:

```
PORTB = PORTB ^ 0x20 // invertovanie, prepnutie, bitu b5
PORTB ^ = 0x20
```

0x12 = 0b0001 0010

OR:

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

0xFC = 0b1111 1100

AND:

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

30

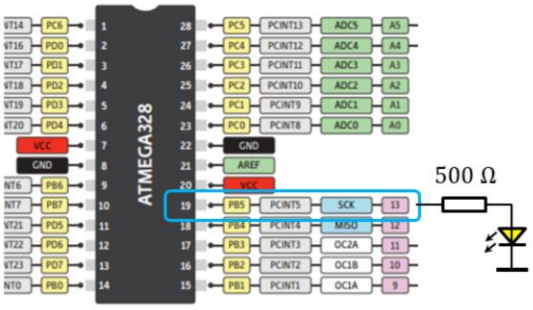
Programovanie I/O bit-ov:

V prostredí ARDUINO UNO môžeme použiť príkazy:

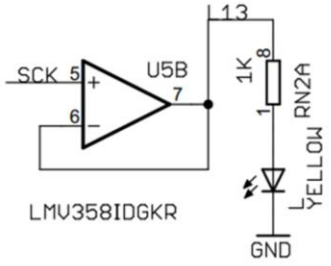
```
// initialize digital pin LED_BUILTIN as an output.
pinMode(LED_BUILTIN, OUTPUT);          // DDRB |= (1 << DDB5);
                =13
// turn the LED on (HIGH is the voltage level)
digitalWrite(LED_BUILTIN, HIGH);       // PORTB |= (1 << PORTB5);

// turn the LED off by making the voltage LOW
digitalWrite(LED_BUILTIN, LOW);        // PORTB &= ~(1 << PORTB5);
```

St. ver.



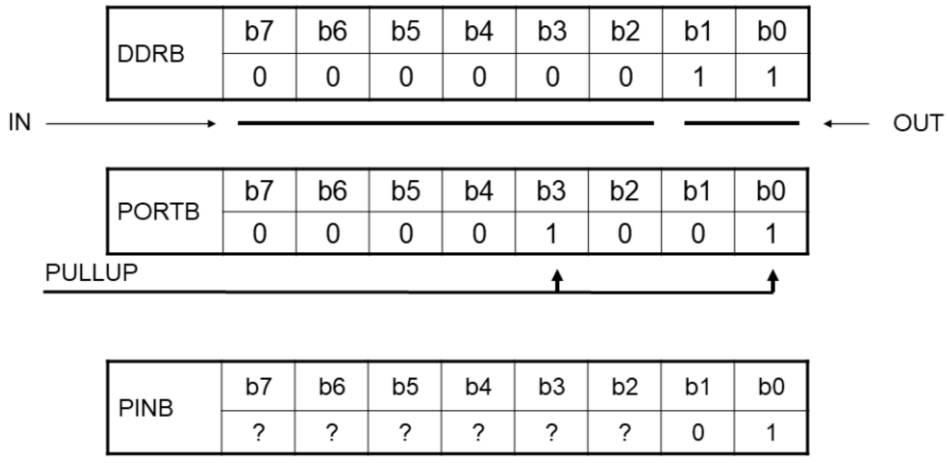
Nov. ver.



31

Na doske ARDUINO UNO máme na PIN 13 (Port B bit 5) pripojenú žltú ledku. Najskôr bit 5 nastavíme ako výstupný. Potom môžeme led-ku rozsvietiť/zhasnúť. Na obr. vľavo je led-ka pripojená cez rezistor na pin portu.

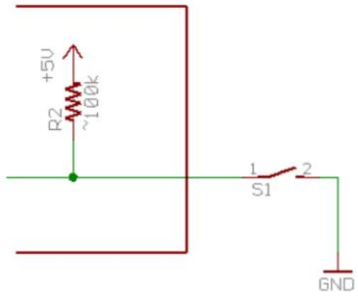
Na obr. vpravo je medzi pin a ledku + rezistor zapojený ako sledovač rail to rail OA LMV358IDGKR.


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

32

I/O Príklad: jednoduchý kontakt

Netreba pripájať žiadne externé pullup rezistory



33

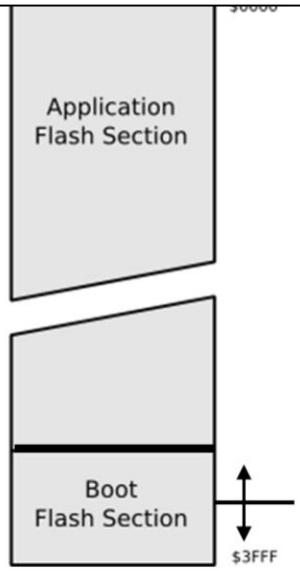
Pamäť programu

- Program je uložený vo FLASH.
  Preprogramovateľná nonvolatile pamäť (nezávislá od výpadku napájania)
- Napájanie: 1,8V až 5,5V
- Stálosť údajov – cca 20rokov
- Počet Write/Erase cyklov – 10000, Bezpečnostné poistky

Flash pamät je rozdelená do dvoch sekcií.

– Prvá sekcia: Blok pamäte určený pre ukladanie aplikačných programov. Do tejto časti pamäte môžeme program zapísať pomocou PROGRAMÁTORA alebo pomocou krátkeho programu zapísaného v 2. sekcii, ktorý prijíma dáta zo sériového portu.

– Druhá sekcia: Po pripojení napájania môže program uložený v tejto pamäti prepísať program v sekcii prvej. Program zapísaný do tejto sekcie sa zvykne nazývať Bootloader.



SPM inštrukcia

Ukladanie údajov do pamäte:

Informácia (Program, Dáta) sa ukladá a vyberá z pamäte po tzv. strojových slovách.
Šírká dátovej zbernice môže vo všeobecnosti biť: 8b = 1B, 16b = 2B, 32b = 4B, 64b = 8B)

Ak sa nezhoduje šírka dátovej zbernice a veľkosť pamäťového miesta, vznikne problém. Ktorý Byte uložiť na ktorú časť pamäťového miesta. Týmto vznikol problém nazývaný ENDIAN-y alebo NUXI.

Pojmy **Big-endian** a **Litle-endian** pochádzajú z Gulliver's Travels od Jonathana Swifta.

Obyvatelia sa rozdelili do dvoch skupín

- Big Endians bola skupina poddaných, ktorá rozbíjala vajíčka na tučnom konci. Boli to vzbúrenci.
- Litle Endians skupina poddaných verná kráľovi, ktorá rozbíjala vajíčko po starom, na tenkom konci.

35

$$(305\,419\,896)_{10} = (12\,34\,56\,78)_H \quad \begin{matrix} 3 & 2 & 1 & 0 \end{matrix}$$

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 ?), zistíme.

36

Problematika Endianov vo všeobecnosti.

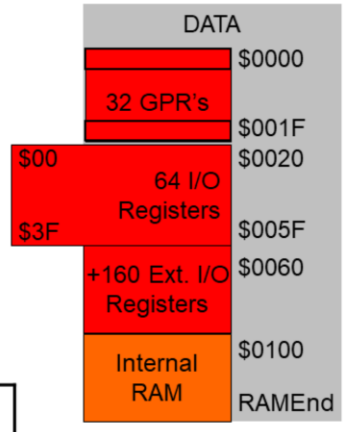
Pamäť dát mikrokontrolerov AVR

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



Na akej adrese končí IRAM MCU AVR 328?

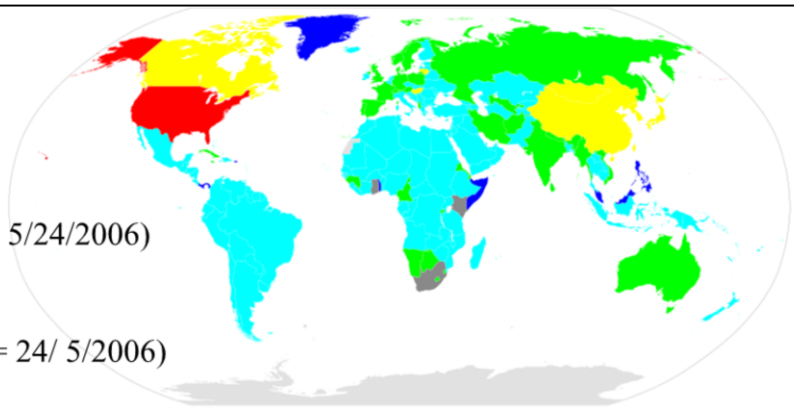
37

AVR procesor môže adresovať až 64KB pamäte dát

- Laod/Store inštrukcie.
- „direct“ alebo „indirect“ adresovanie
- Pri nepriamom adresovaní, sa použije časť GPR ako smerníky na adresu.
- ďalšie adresovacie módy: posunutie, post-increment or pre-decrement.

AVR 328 má 2KB (2048B) internej SRAM.

Date format by country - Wikipedia



Dátum:
USA : middle - endian
Month, Day, Year (May, 24th,2006 = 5/24/2006)

Europe : little – endian (prevážne)
Day, Month, Year (24th, May, 2006 = 24/ 5/2006)

China, Japan & ISO 8601: big - endian
Year, Month, Day (2006, May, 24th = 2006-05-24)

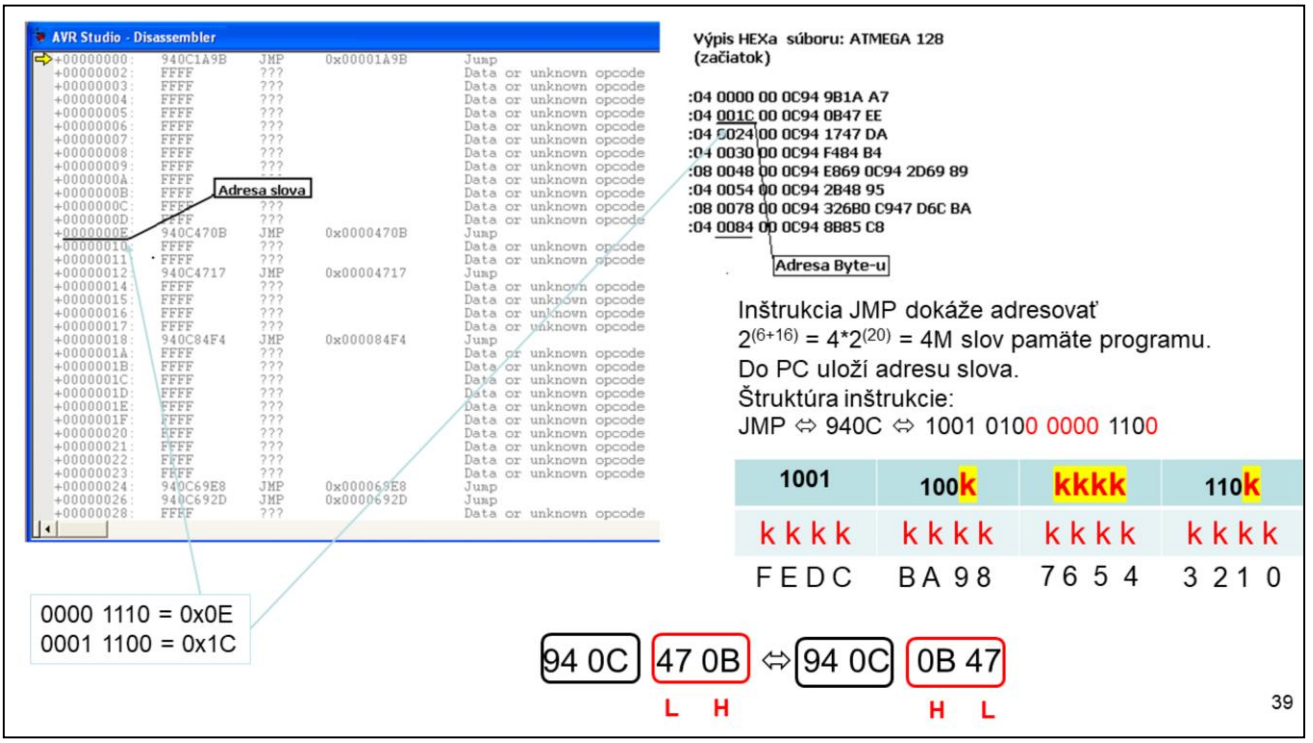
Poznámky k endianom:

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 vykonať transformáciu.

38

Západná a stredná Európa sú síce označené dvoma rôznymi farbami, ale majú spoločný jeden spôsob ukladania.



Výsledkom prekladu pre konkrétny typ mikrokontrolera je HEXA súbor, ktorý má jasnú štruktúru. Program sa do pamäte programu ukladá po slovách. Preto je v AVR štúdiu napr. adresa 0xE, ale v HEX súbore tomu odpovedá adresa bytu 0x1C.

Možno povedať, že ešte trocha komplikovanejšie je to keď chceme napísať vlastný bootovací program.

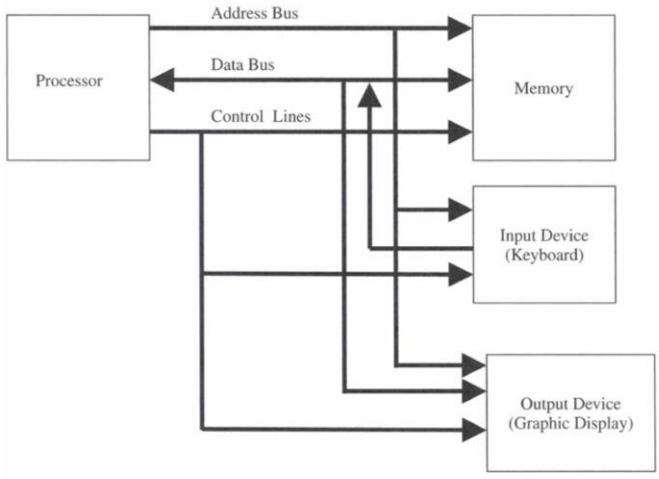
Tu a na tomto mieste nebudeme odpovedať na otázku prečo taká zložitá štruktúra kódu operácie.

AVR data sú ukladané do oboch pamätí RAM (pamäť dát) aj FLASH (pamäť programu) v LE, a to aj napriek tomu, že pamäť programu (FLASH) je organizovaná v BE.

Mapovanie I/O obvodov, I/O Mapping

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

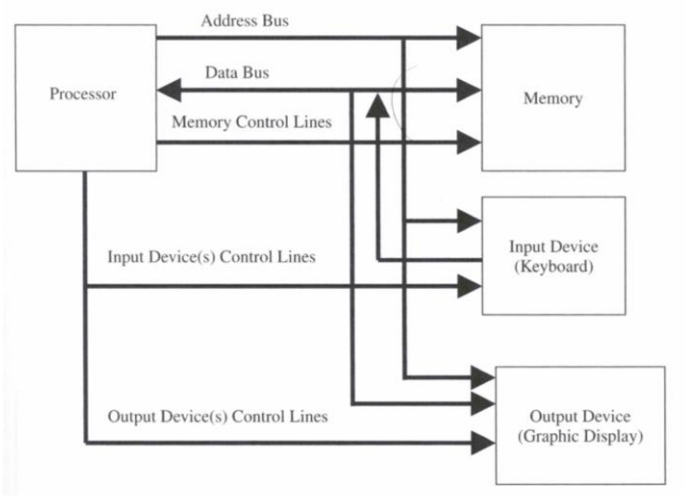
- **Pamäťovo mapované I/O** Pre I/O zariadenia možno použiť celú pamäť
  - Periférie a pamäť zdieľajú ten istý pamäťový priestor
  - Žiadny špeciálne príkaz pre I/O
  - (Motorola).



40

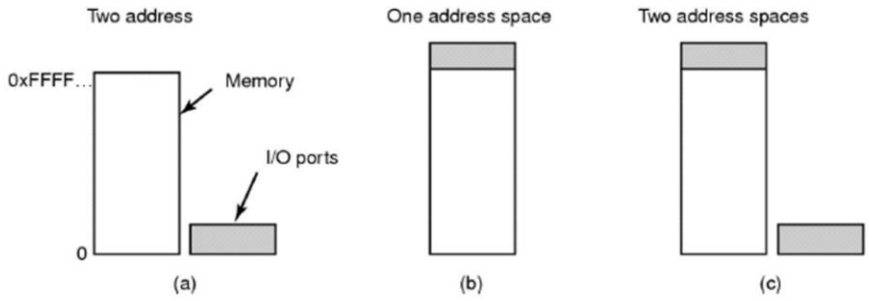
Mapovanie I/O obvodov, I/O Mapping

- **Samostatné mapované I/O zariadenia**
  - Oddelený adresný priestor
  - Špeciálne vodiče pre výber I/O a pamäť
  - Špeciálne príkazy I/O
  - (Intel)



41

Mapovanie I/O obvodov, I/O Mapping

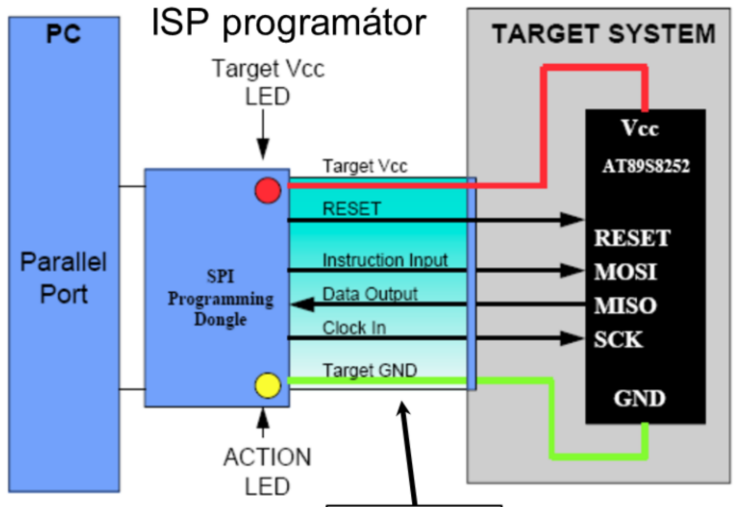


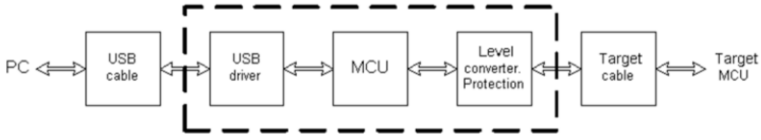
- Separate I/O and memory space PC
- Memory-mapped I/O 8051
- Hybrid AVR

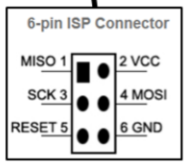
Programovanie AVR mikrokontrolerov:

Paralelné:
Vyžaduje špeciálny programátor. Je rýchle.

Sériové:
Nevyžaduje špeciálny programátor. Môžeme programovať zabudovaný obvod pomocou 6-tich vodičov.
"**In-System Programming** (**ISP**)".

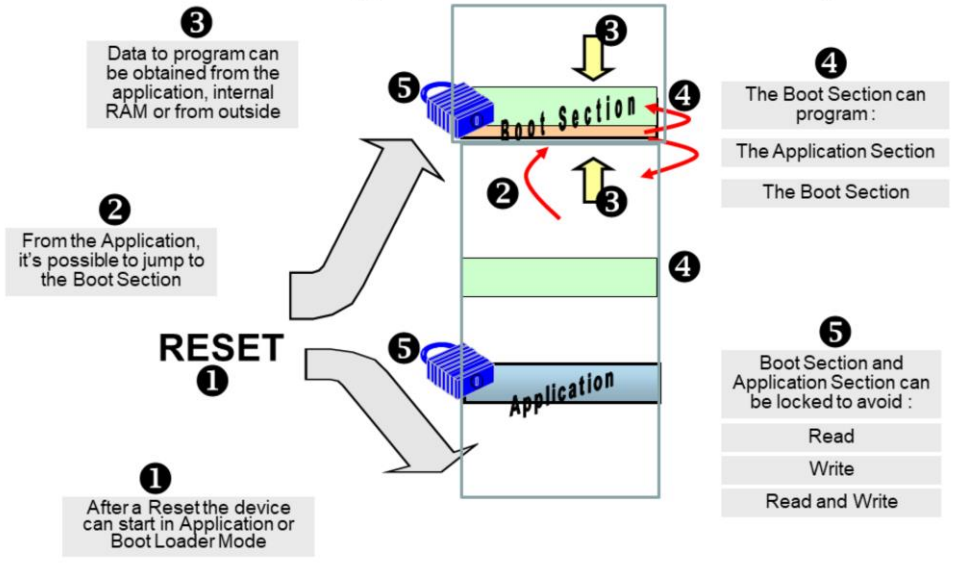






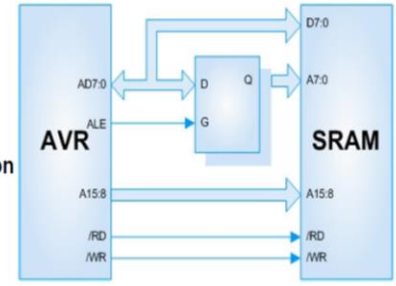
43

Self Programming Security



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



45

Niektoré mikrokontrolery majú vyvedené zbernice cez porty. To znamená, že môžem pripojiť externú pamäť RAM a do priestoru externej pamäte môžu biť mapované externé periférie.

High Level Languages

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

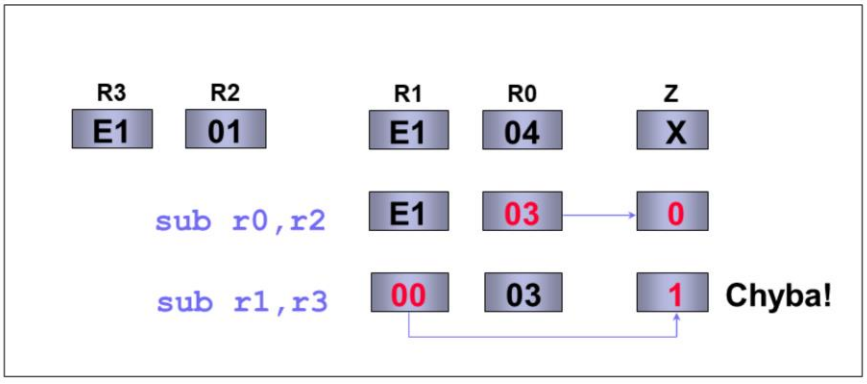
- RISC procesor.
  počet inštrukcií nie je redukovaný za každú cenu, viď. ďalší príklad

- K dispozícii máme 32 GPR registrov, ktoré sú vo funkcii akumulátorov.

Rozdieľ dvoch 16-Bit-ových čísiel

Without Zero Flag Propagation

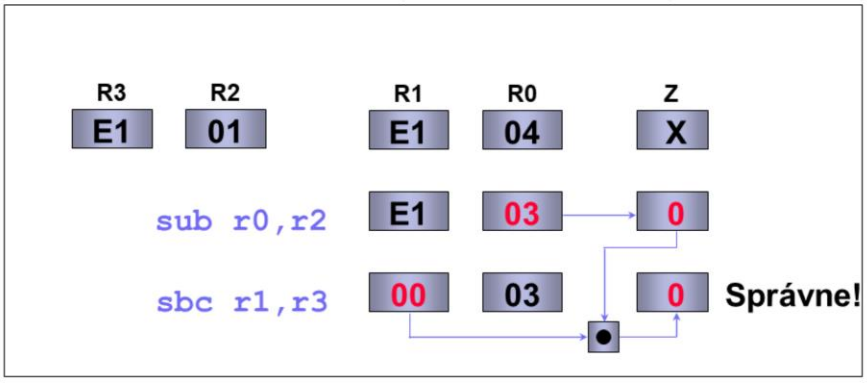
R1:R0 - R3:R2 (0xE104 – 0xE101)



Rozdieľ dvoch 16-Bit-ových čísiel

With Zero Flag Propagation

R1:R0 - R3:R2 (0xE104 – 0xE101)



Na druhej strane treba uviesť, že niekedy redukovali počet inštrukcií dôsledne.
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 záporными 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“ $-65_{10} * 2_{10} = -130_{10} + 256_{10} = +126_{10}$

49

AVR Assembler User Guide

Mnemonics	Operands	Description	Operation	Flags	#Clock Note
BIT AND BIT-TEST INSTRUCTIONS					
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
SWAP	Rd	Swap Nibbles	Rd(3..0) ↔ Rd(7..4)	None	1
BSET	s	Flag Set	SREG(s) ← 1	SREG(s)	1
BCLR	s	Flag Clear	SREG(s) ← 0	SREG(s)	1
SBI	P, b	Set Bit in I/O Register	I/O(P, b) ← 1	None	2

TYPE	SIZE (bytes)	RANGE
`unsigned char`	1	0..255
`signed char`	1	-128..127
`char (*)`	1	0..255
`unsigned short`	2	0..65535
`(signed) short`	2	-32768..32767
`unsigned int`	2	0..65535
`(signed) int`	2	-32768..32767

TYPE	SIZE (bytes)	RANGE
`pointer`	2	0..65535
`unsigned long`	4	0..4294967295
`(signed) long`	4	-2147483648..2147483647
`float`	4	+/-1.175e-38..3.40e+38
`double`	4	+/-1.175e-38..3.40e+38

`(*) char` is equivalent to `unsigned char`.

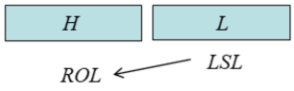
`floats` and `doubles` are in IEEE standard 32-bit format with 8-bit exponent, 23-bit mantissa, and 1 sign bit.

50

Celočíselná aritmetika so znamienkom (word):

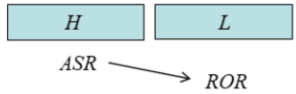
V nasledovnom príklade sú použité inštrukcie:

X =X<<1



a

X =X>>1



Prečo v prvom príklade nie je ASL namiesto ROL?

Príklady: 4/2 =2, 3/2= 1. Koľko je (-3)/2 ?

51

```
@0000005D: main
45:        init_devices();
+0000005D:   DFE6        RCALL   PC-0x0019       Relative call subroutine
47:        cis1=-2733;
+0000005E:   E563        LDI     R22,0x53        Load immediate
+0000005F:   EF75        LDI     R23,0xF5        Load immediate
48:        cis2=2733;
+00000060:   EA4D        LDI     R20,0xAD        Load immediate
+00000061:   E05A        LDI     R21,0x0A        Load immediate
49:        cis1=cis1<<1;
+00000062:   0F66        LSL     R22             Logical Shift Left
+00000063:   1F77        ROL     R23             Rotate Left Through Carry
50:        cis2=cis2>>1;
+00000064:   9555        ASR     R21             Arithmetic shift right
+00000065:   9547        ROR     R20             Rotate right through carry
51:        aa: goto aa;
+00000066:   CFFF        RJMP    PC-0x0000       Relative jump
+00000067:   9508        RET                     Subroutine return
```

H (R23)	L (R22)
0xF5 =1111 0101	0x53=0101 0011
	$<L>_{LSL}$=1010 0110, <C>=$b7_H$=0
$<H>_{ROL}$=1110 101<C>, <C>=$b7_H$=1	
0xEA	0xA6

-2733d * 2 =

-5466d

H (R23)	L (R22)
0xF5 =1111 0101	0x53=0101 0011
$<H>_{ASR}$=1111 1010, <C>=$b0_H$=1	
	$<L>_{ROR}$=1010 1001, <C>=$b0_H$=0
0xFA	0xA9

-2733d / 2 = ? -1366 ?

? -1365d ?

52