Proiectare cu Microprocesoare

Curs 1

Sisteme cu microprocesoare. Microcontrolere AVR.
Introducere in Arduino.

An 3 CTI, Seria A
Semestrul I

Lector: Răzvan Itu

Introducere

Obiective

- Cunoaşterea, înţelegerea şi utilizarea conceptelor: microprocessor, magistrală, memorie, sistem de intrare-ieşire, metode de transfer a datelor, interfeţe.
- Analiza şi proiectarea sistemelor cu microprocesor

Cunoştinţe preliminare necesare

- Proiectare Logică, Proiectarea sistemelor numerice, Arhitectura Calculatoarelor, Programare în Limbaj de Asamblare, Programarea Calculatoarelor (C/C++)

Structura disciplinei

- 2C + 1L + 1P / săptămână

Structura cursului

- Partea 1 – ATMEL (ATmega2560, Arduino) şi aplicaţii
- Partea 2 – ESP32 şi aplicaţii

Tematica laboratorului

- Lucrări practice folosind plăci Arduino (ATmega2560 (MEGA2560), ATmega328P(UNO)), ESP32 şi multiple module periferice

Bibliografie

Slide-urile de curs, disponible pe site:

http://users.utcluj.ro/~razvanitu => Teaching

Microcontrollere

G. Grindling, B. Weiss, Introduction to Microcontrollers, Vienna Institute of Technology, 2007.

https://ti.tuwien.ac.at/ecs/teaching/courses/mclu/theory-material/Microcontroller.pdf

Atmel AVR, Arduino

M. A. Mazidi, S. Naimi, S. Naimi, The AVR Microcontroller and Embedded Systems Using Assembly And C, 1-st Edition, Prentice Hall, 2009.

Michael Margolis, Arduino Cookbook, 2-nd Edition, O'Reilly, 2012.

ESP32

N. Kolban, Kolban's Book on ESP 32, 2017

Documente suplimentare

Data sheets Atmel, Intel etc, tuturiale Arduino: **http://arduino.cc/en/Tutorial/HomePage**

ESP 32 datasheet:
https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

Evaluare

Evaluare: nota examen (E) + nota laborator/proiect (LP)

```
if (LP > = 5) AND (E > = 4.5)
      Final_mark = 0.5 *LP + 0.5 * E
else
     Final_mark = 4 OR Absent
```

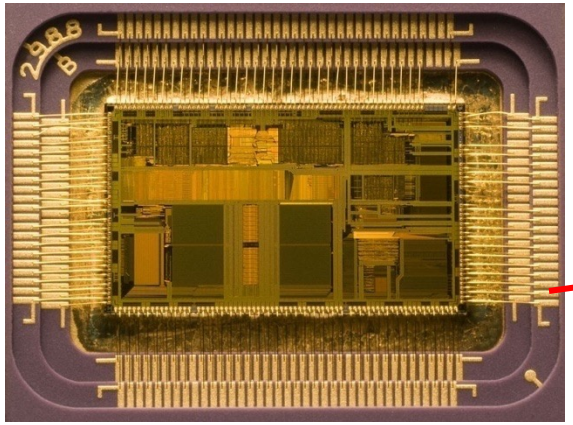
Bonus - se poate acorda pentru activitate deosebită la curs/ laborator, sau pentru participarea la concursuri studenţeşti

Ce este un microprocesor

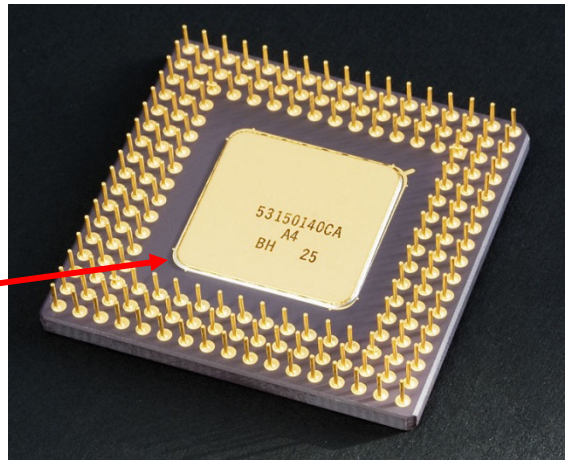
Un **microprocesor** incorporeaza toate sau majoritatea functiilor unei **unitati centrale de procesare** intr-un singur circuit integrat.

O unitate centrala de procesare **(Central Processing Unit, CPU)** este o masina logica ce poate executa programe de calculator.

Functia fundamentala a oricarui CPU, indiferent de forma fizica pe care o are, este sa execute o **secventa de instructiuni (programul)**, stocate intr-o memorie. Executia instructiunilor se face de obicei in patru pasi: citire instructiune (**fetch**), decodificare (**decode**), executie (**execute**) si scriere rezultate (**write back**).



Intel 80486DX2 , interior



Intel 80486DX2 – vedere exterioara

Scurta istorie

Microprocesoare pe 4 biti: Intel's 4004 (1971), Texas Instruments (TI) *TMS* 1000, si Garrett AiResearch's Central Air Data Computer (CADC).

Microprocesoare pe 8 biti: 8008 (1972), primul procesor pe 8 biti. A fost urmat de Intel 8080 (1974), Zilog Z80 (1976), si alte procesoare derivate pe 8 biti de la Intel. Competitorul Motorola a lansat Motorola 6800 in August 1974. Arhitectura acestuia a fost clonata si imbunatatita in MOS Technology 6502 in 1975, cu popularitate similara lui Z80 in anii 1980.

16 bit (Intel 8086, 80186, 80286, 8086 SX, TMS 9900)

32 bit (MC68000, Intel 80386DX, 80486DX, Pentium, MIPS R2000 (1984) si R3000 (1989) etc.)

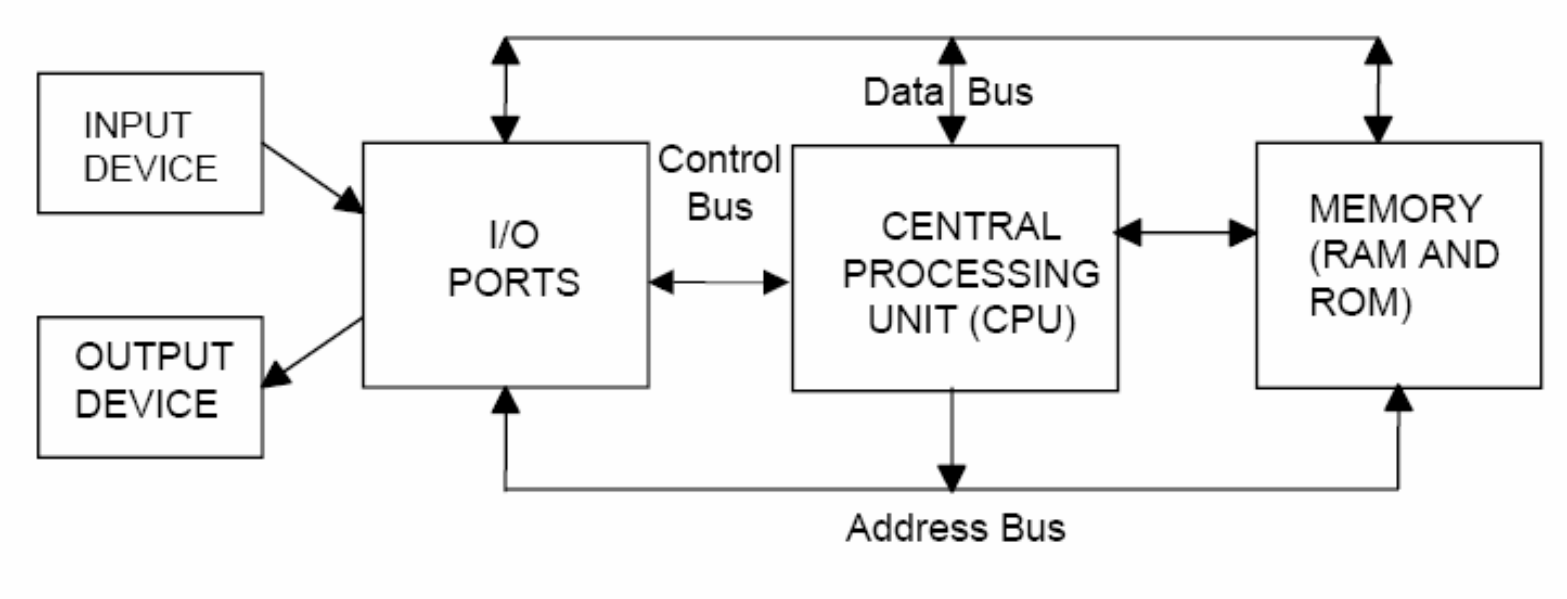
64 bit (majoritatea procesoarelor moderne)

Tipuri:

RISC: MIPS (R2000, R3000, R4000), Motorola 88000, AVR

CISC: VAX, PDP-11, Motorola 68000, Intel x86

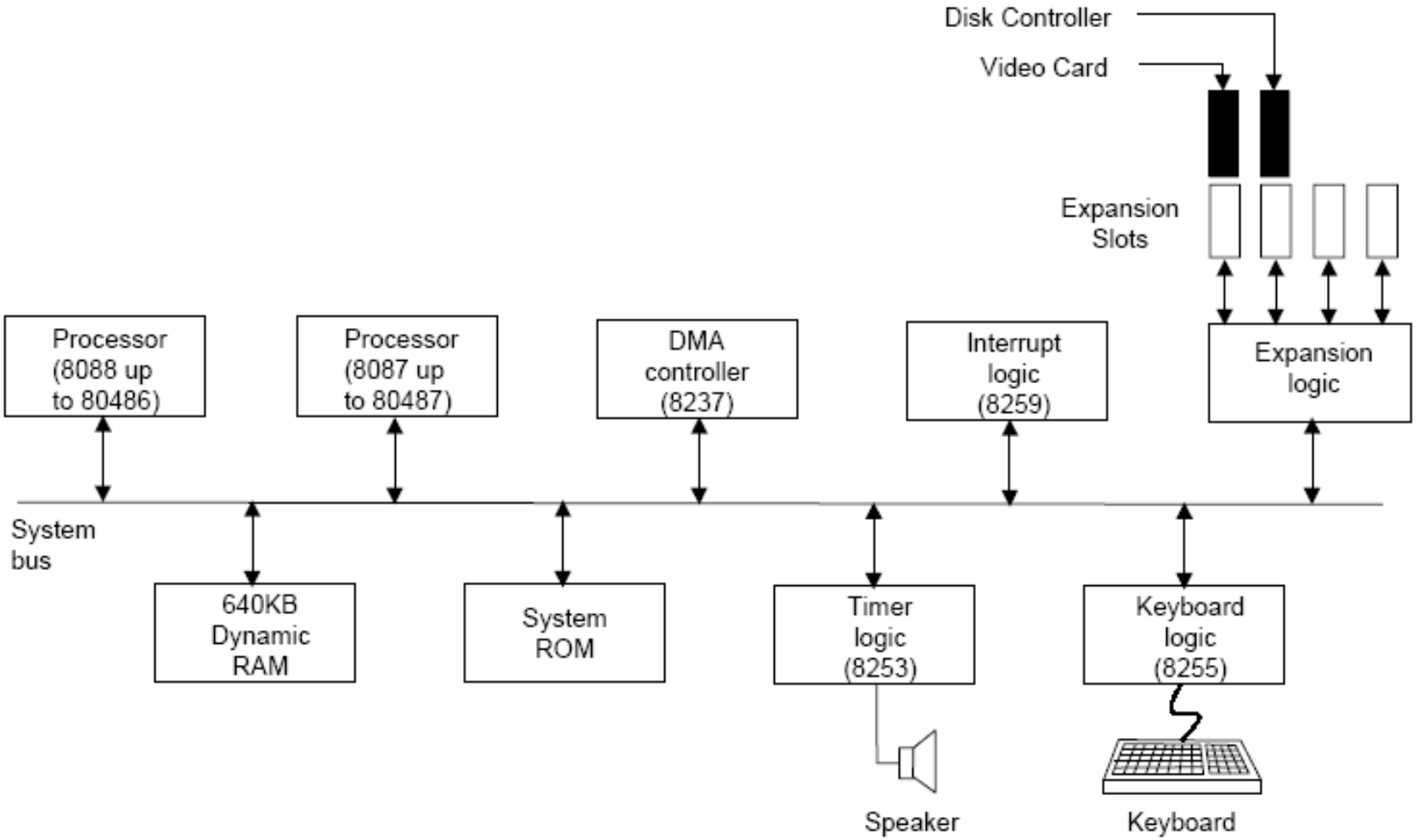
Sisteme cu microprocesor



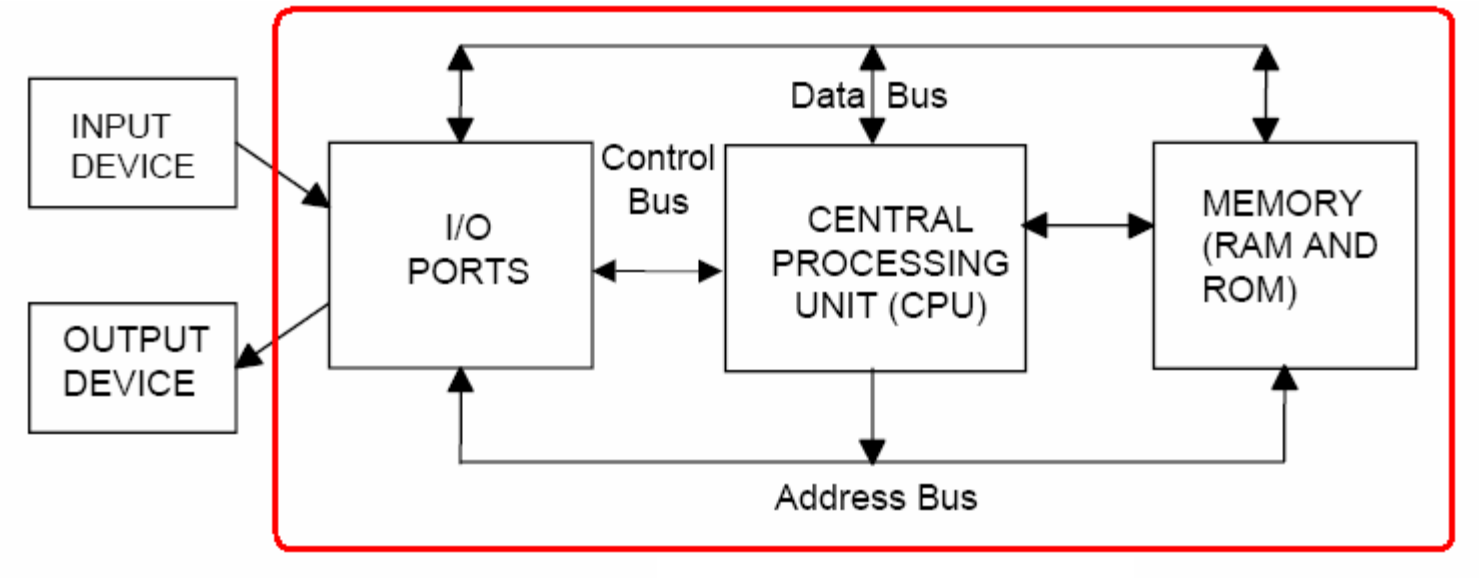
Dispozitive esentiale: CPU, Memorie, I/O

Dispozitive aditionale: Controller intreruperi, DMA, coprocesor, etc

Exemplu: placa de baza PC



Microcontroler (MCU)



Multiple componente ale unui sistem cu microprocesor sunt incluse in acelasi circuit integrat - Microcontroler

- Memorie RAM si ROM (Flash), pentru program si date
- Unele dispozitive periferice (Timer, Numarator, Controllere pentru comunicatii seriale / paralele, etc)

Proiectarea cu Microprocesoare

Obiectiv general: utilizarea microprocesoarelor (microcontrollerelor) in dezvoltarea de sisteme electronice adaptate unor probleme specifice.

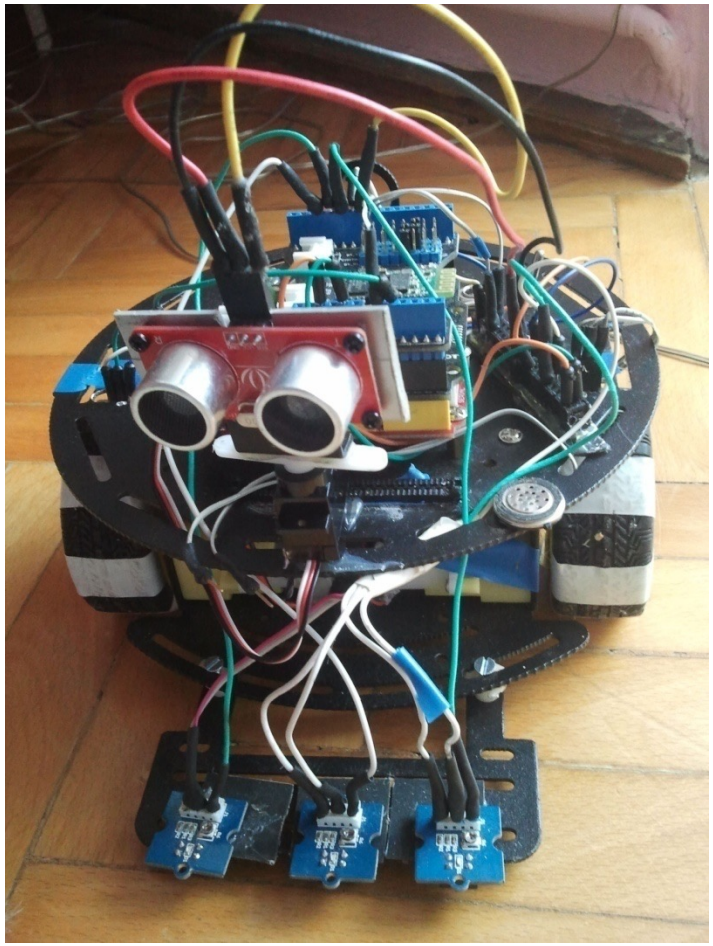
Exemple de aplicatii: roboti autonomi, senzori inteligenti, senzori mobili, procesare de semnal audio, procesare de imagini, controlul automat al unor procese, etc.

Pasii pentru indeplinirea acestui obiectiv:

- Studiul capabilitatilor microcontrollerului, familiarizarea cu programarea acestuia;
- Studiul resurselor integrate in microcontroller si resurselor disponibile pe placa cu microcontroller;
- Studiul dispozitivelor externe necesare pentru rezolvarea unor probleme specifice;
- Studiul interfetelor de comunicare, a formatului datelor, si a diagramelor de timp, necesare pentru conectarea microcontrollerului la dispozitivele externe.

Proiectarea cu Microprocesoare

Exemplu: proiectarea unui robot capabil sa se deplaseze autonom, evitand obstacolele, sau sub controlul unui operator uman, sau ghidat de o banda de culoare inchisa.



Microcontroller: AVR ATMega328, placa Arduino, programare in C/C++

Componente interne: porturi I/O, intreruperi, interfata de comunicare seriala, generator PWM

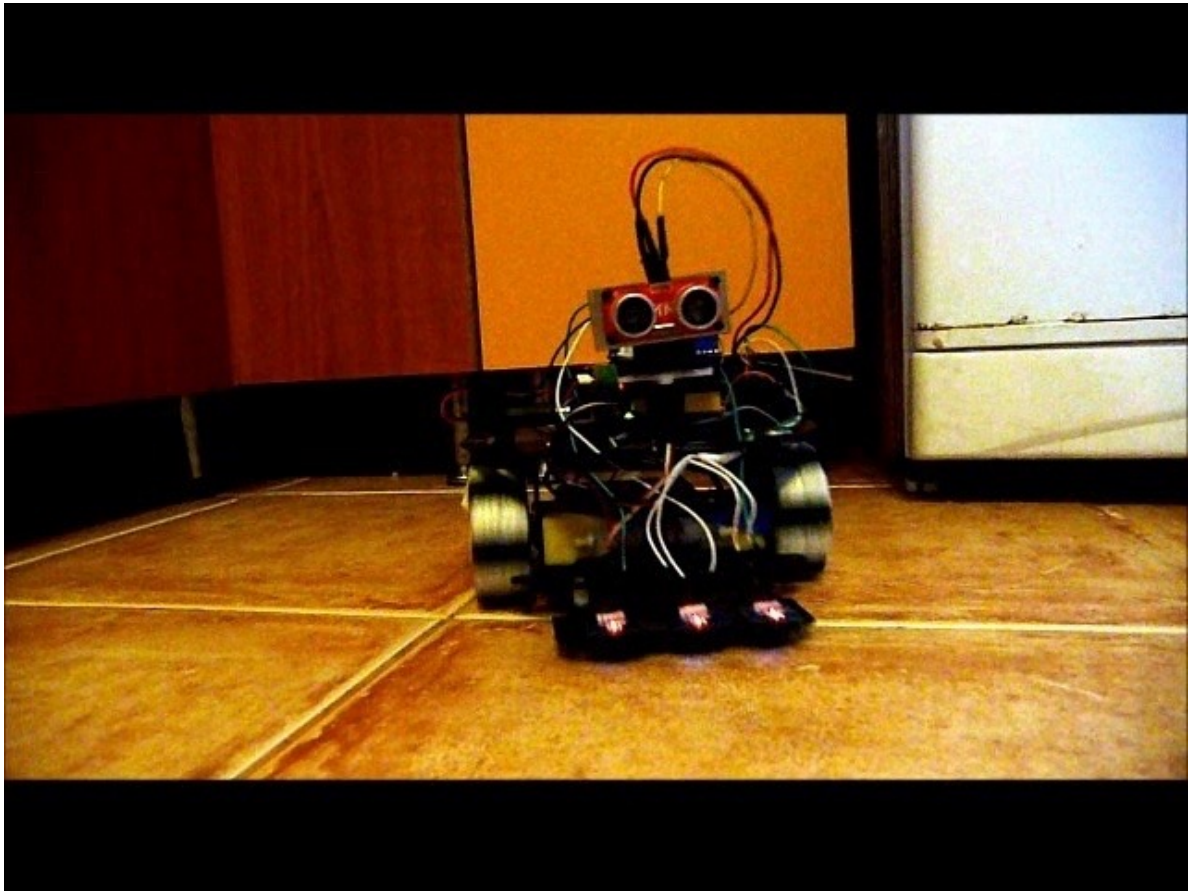
Componente externe: motoare DC, 1 motor servo, senzori de reflectivitate, punte H, senzor de distanta sonar, modul de comunicare Bluetooth.

Comunicare: seriala, tip UART, intre MCU si modulul Bluetooth, PWM intre MCU si servo, si intre MCU si puntea H, semnal analogic de la senzorii de reflectivitate, puls digital intre sonar si MCU.

Algoritmi: scanare mediu si detectia obstacolelor, urmarirea liniei, controlul rotilor pentru mersul in linie dreapta, etc.

Proiectarea cu Microprocesoare

Exemplu: proiectarea unui robot capabil sa se deplaseze autonom, evitand obstacolele, sau sub controlul unui operator uman, sau ghidat de o banda de culoare inchisa.

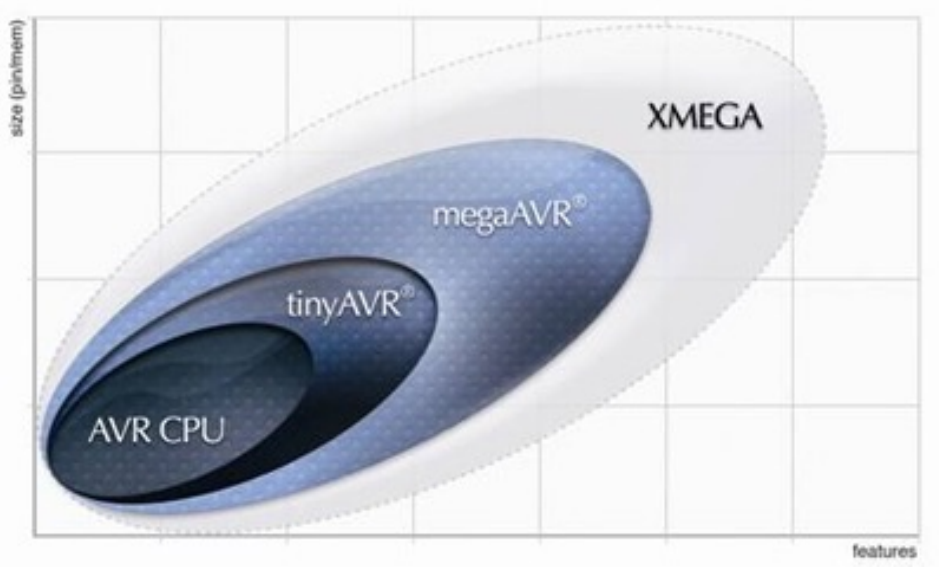


Proiectarea cu Microprocesoare

Exemplu: proiectarea unui robot capabil sa se deplaseze autonom, evitand obstacolele, sau sub controlul unui operator uman, sau ghidat de o banda de culoare inchisa.

Familia de microcontrolere Atmel AVR 8 biti

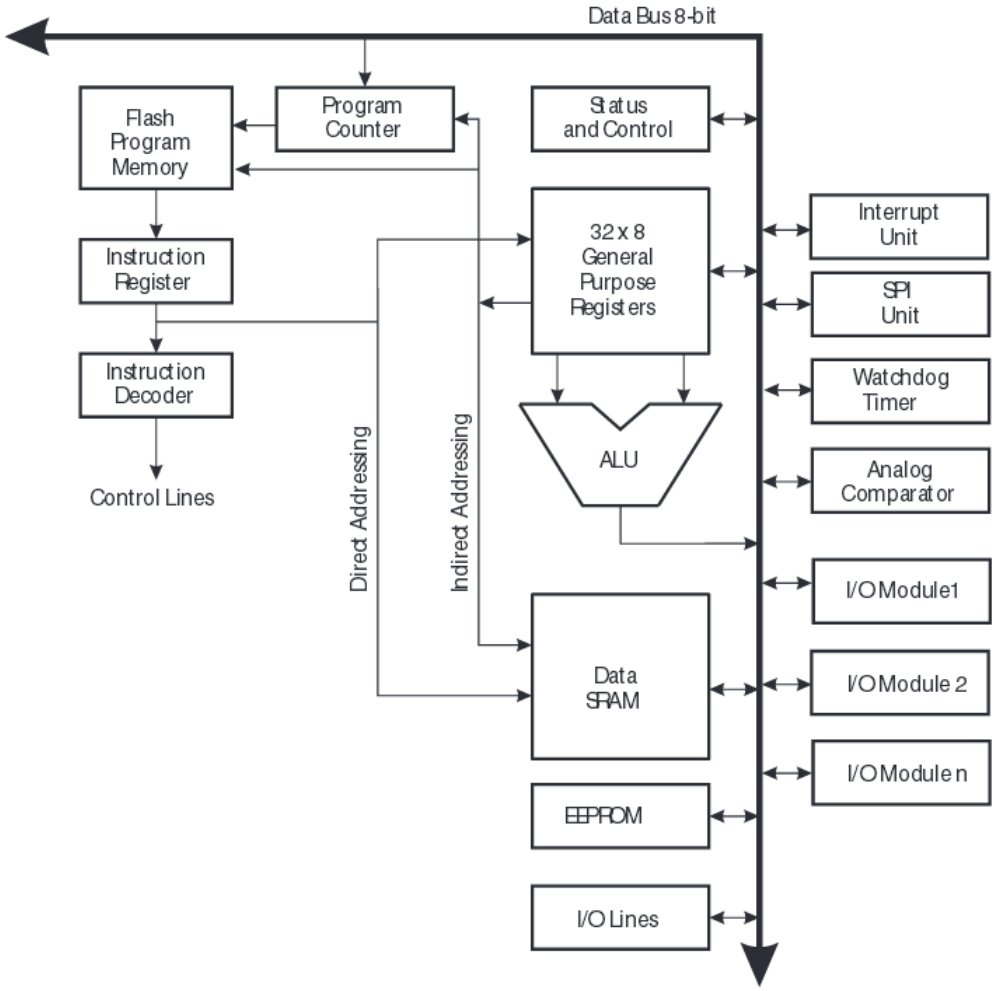
- Arhitectura RISC
- Executie 1 instructiune / ciclu
- 32 registri de uz general
- Arhitectura Harvard
- Tensiune de alimentare 1.8 - 5.5V
- Frecventa controlata software
- Mare densitate a codului
- Gama larga de dispozitive
- Numar de pini variat
- Compatibilitatea integrala a codului
- Familii compatibile intre pini si capabilitati
- Un singur set de unelte de dezvoltare



tinyAVR
1–8 kB memorie program
megaAVR
4–256 kB memorie program
Set extins de instructiuni (inmultire)
XMEGA
16–384 kB memorie program
Extra: DMA, suport pentru criptografie
AVR specific pentru aplicatii
megaAVR cu interfete particulare:
LCD, USB, CAN etc.

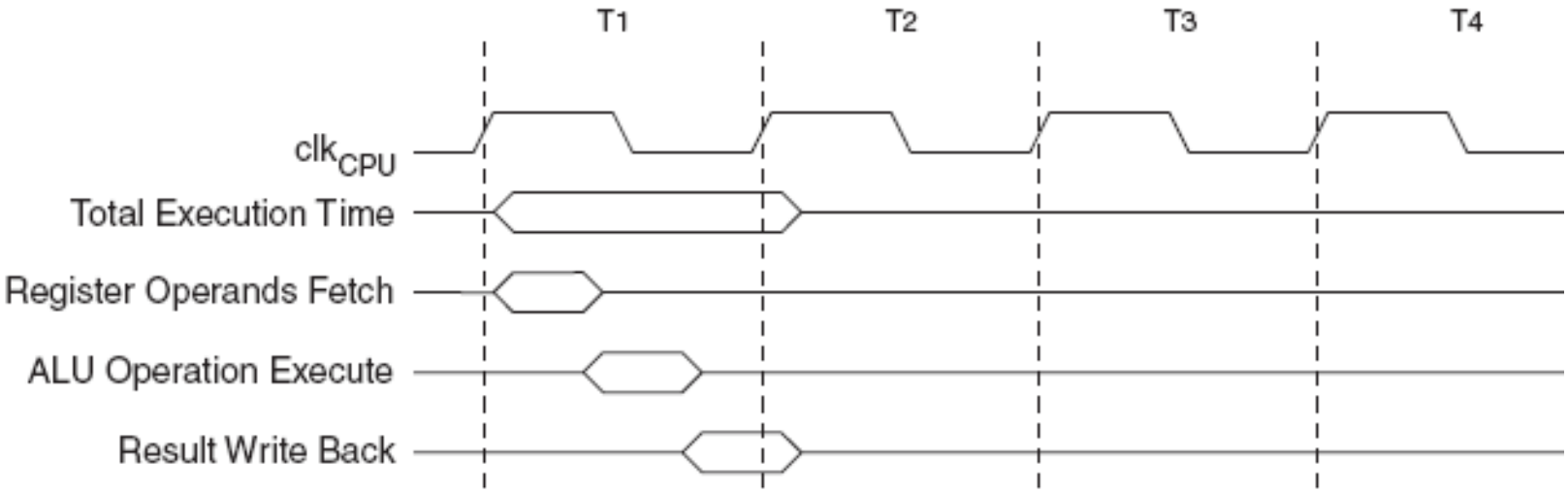
Arhitectura generala a unui microcontroller AVR

- Masina RISC (Load-store cu doua adrese)
- Arhitectura Harvard modificata – exista instructiuni speciale care pot citi datele din memoria program
- Pipeline pe doua nivele: Fetch & Execute

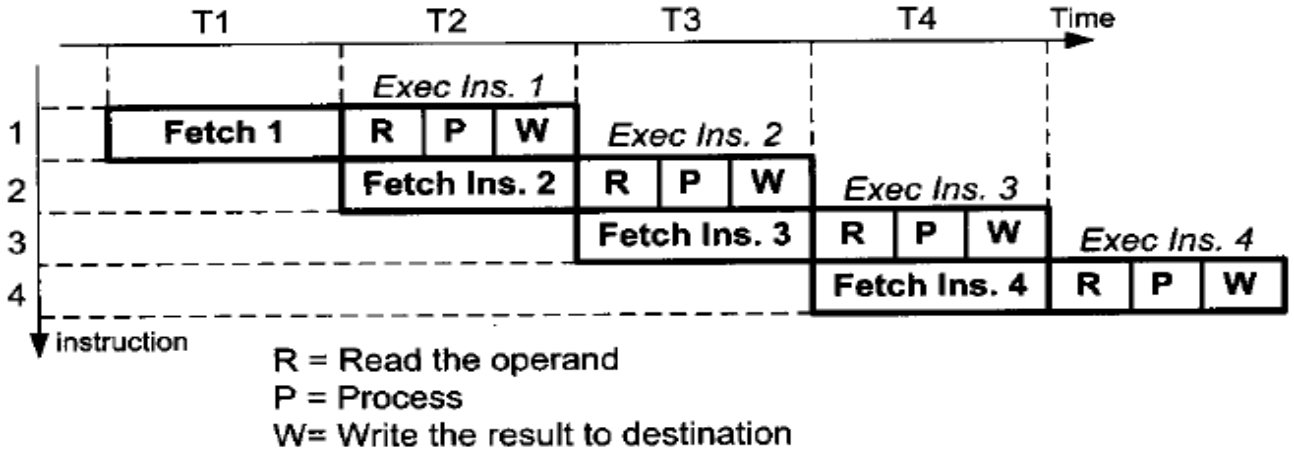


Diagrame de timp AVR

- Executia instructiunilor aritmetico-logice

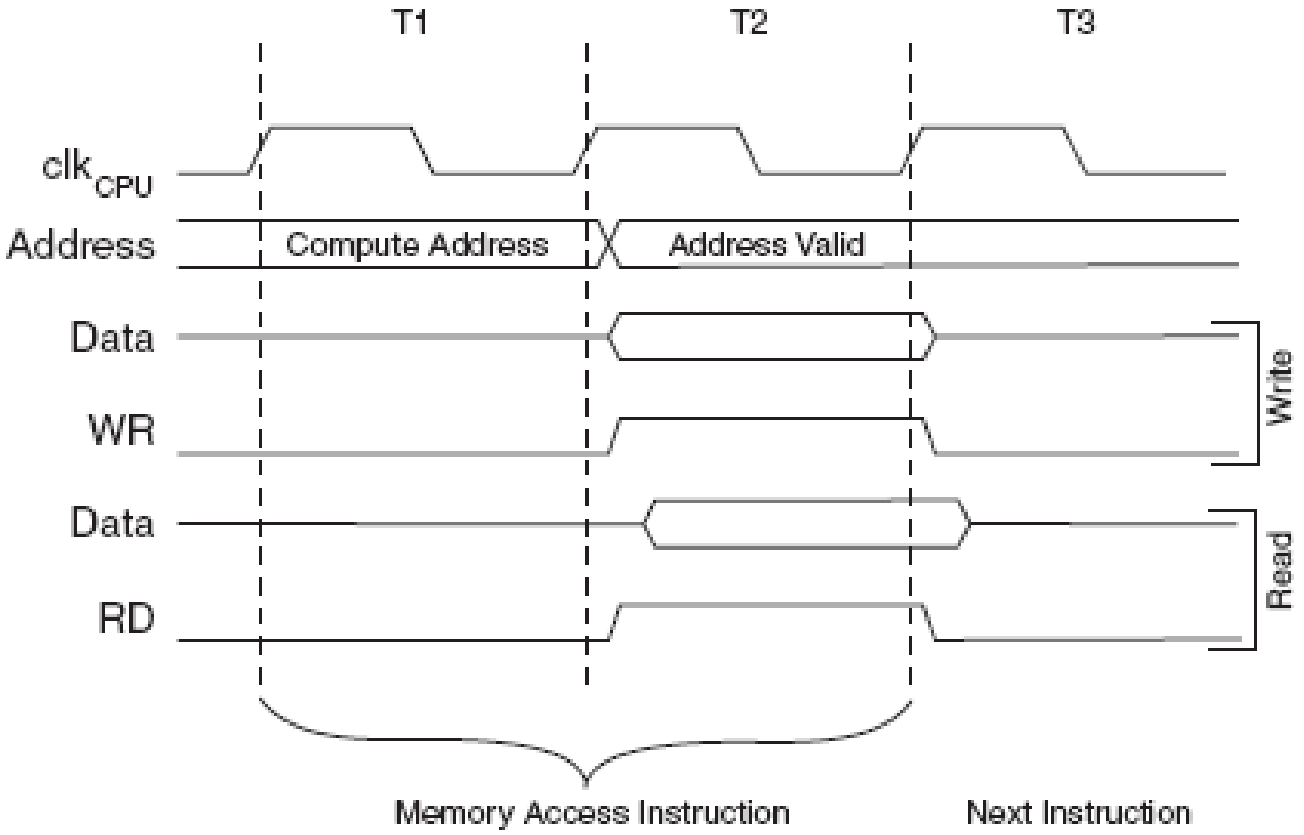


- Pipeline asigura suprapunerea citirii urmatoarei instructiuni cu executia celei curente



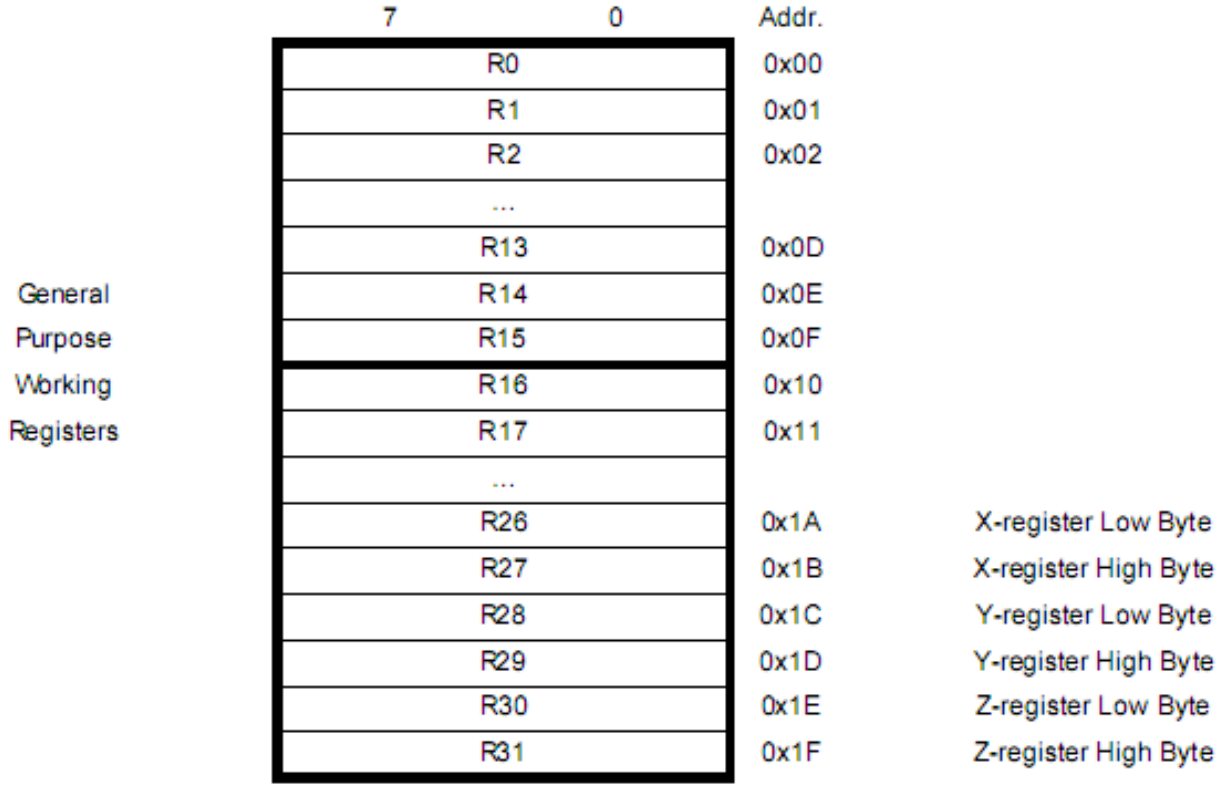
Diagrame de timp AVR

- Instructiunile care acceseaza memoria interna SRAM
  - 2 cicli / instructiune



Registri de uz general (General Purpose Registers – GPR)

- Valori imediate se pot incarca doar in registrii R16-R31
- Registrii R26 – R31 sunt folositi in perechi ca si pointeri
- Fiecare registru are si o adresa in spatiul memoriei de date – adresare uniforma



Operatii cu registri

- Copiere date

  mov r4, r7

- Lucrul cu valori immediate – posibil doar cu registrii r16 – r31

  ldi r16, 5
  ori r16, 0xF0
  andi r16, 0x80
  subi r20, 1

- Operatii aritmetice si logice intre registri

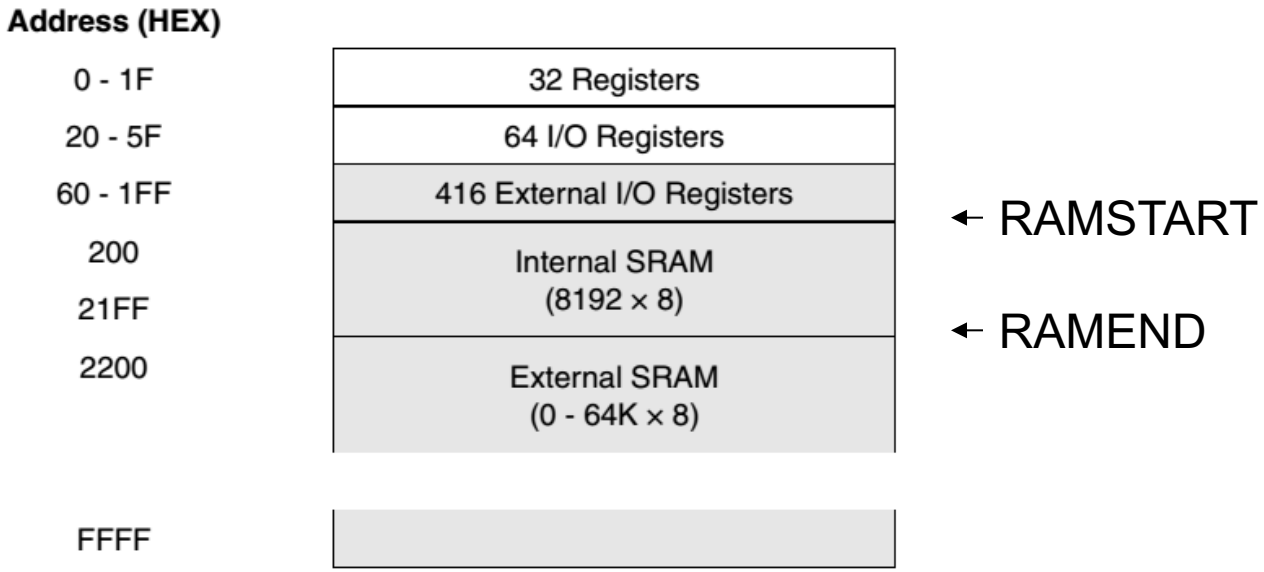
  add r1, r2
  or r3, r4
  lsl r5
  mul r5, r18 – r1:r0 = r5*r18
  rol r7
  ror r9
  inc r19
  dec r17

Memoria de date

- Primele 32 de adrese – blocul de registri
- 64 de adrese – registri I/O accesabili prin instructiuni speciale
- 100+ adrese – spatiu I/O extins, accesabil prin instructiuni standard de acces la memorie
- SRAM, de ordinul Kbytes (2, 4, 8 …)
- Posibilitate de extensie pana la 64 KB

Constantele predefinite RAMSTART si RAMEND marcheaza sfarsitul memoriei interne de date SRAM

ATmega 2560 data memory map



Operatii cu memoria de date

- Accesarea directa a memoriei de date

```
lds r3, 0x10FE
lsl r3
sts 0x10FE, r3
```

- Accesarea indirecta a memoriei de date, prin intermediul registrilor X, Y, Z

```
ldi r27, 0x10      octetul superior al lui X este r27
ldi r26, 0xFE      octetul inferior al lui X este r26
ld r0, X
lsl r0
st X, r0
```

- Accesarea cu autoincrementare/decrementare a adresei

```
ld r0, X+          se acceseaza locatia X, apoi se incrementeaza X
ld r0, +X          se incrementeaza X, apoi se acceseaza locatia X
ld r0, X-
ld r0, -X
```

Memoria program

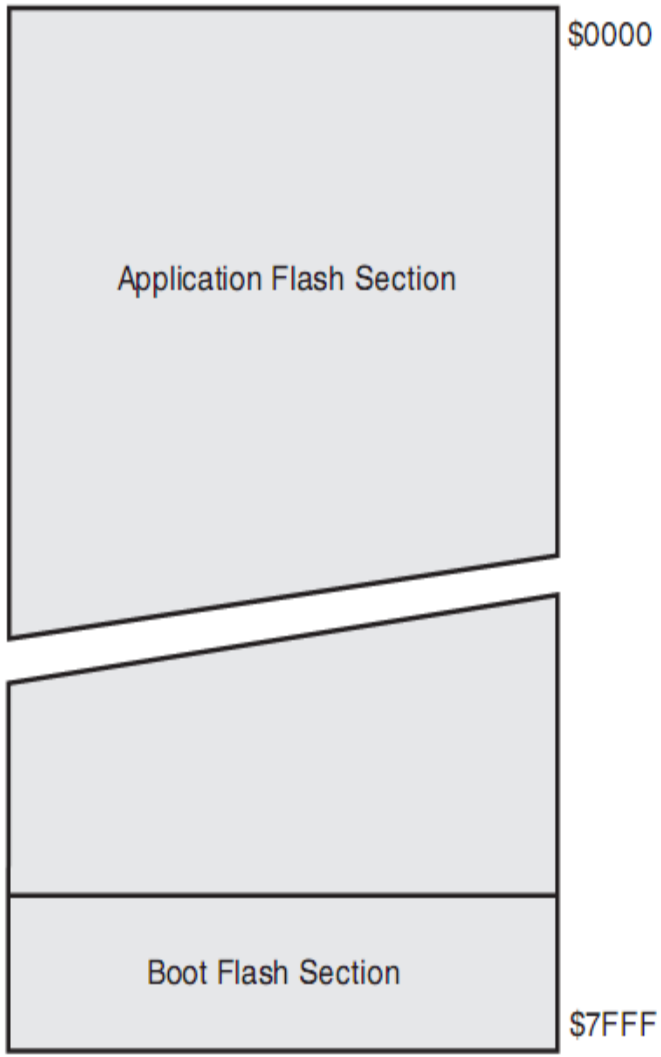
- Memorie flash, pentru programarea aplicatiilor
- Organizata ca in cuvinte de 16 biti
- Doua sectiuni: Boot si Aplicatie
- Cel putin 10000 cicluri scriere/stergere
- Constante pot fi declarate in segmentul de cod, ele vor fi stocate in memoria program
- Accesarea memoriei program
  Citirea - accesul este la nivel de BYTE, adresarea se face doar prin pointerul Z

  LPM r5, Z
  LPM r5, Z+
  LPM r0 este destinatie, Z adresa

ELPM foloseste adresarea extinsa RAMPZ:Z, pentru a accesa memoria program mai mare de 64 KB.

- **Scrierea** – doar pe cuvant

  SPM PM(Z) <= R1:R0



Registrul de stare SREG

- Registrul SREG (8 biti) contine informatii despre starea sistemului si rezultatul unor operatii
- Folosit pentru modificarea comportamentului programului sau pentru salturi conditionate
- Nu este salvat automat la apelul procedurilor sau la executia intreruperilor !

Bit	7	6	5	4	3	2	1	0	
0x3F (0x5F)	I	T	H	S	V	N	Z	C	SREG
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 – activarea globala a intreruperilor
- T – bit de transfer, poate fi copiat prin instructiunile BLD si BST din alt registru
- H – carry intre jumatati de octet, folositor pentru operatii BCD
- S – N xor V, indicator de semn
- V – indicator de overflow, indica schimbarea bitului de semn prin depasire
- N – indicator de rezultat negativ
- Z – indicator al unui rezultat nul
- C - carry

Instructiuni de salt

- Salturi neconditionate
  RJMP – salt relativ, PC +- 2KB
  JMP – salt absolut
  IJMP – salt indirect, la adresa indicata de pointerul Z
- Salturi conditionate
  CP, **CPI** – compara doua numere
  BREQ – salt daca flagul Z este setat (numerele comparate sunt egale)
  BRNE – salt daca numerele nu sunt egale
  BRCS – salt daca flagul C este setat
  SBRS – sare peste urmatoarea instructiune daca un bit intr-un registru e setat
  SBRS r5, 2 – daca bitul 2 din reg. 5 este setat, executa saltul
  SBRC, SBIS, SBIC
- Apeluri de procedura
  RCALL, CALL, ICALL - se salveaza adresa de revenire in stiva, nu se salveaza nimic altceva

- Revenire din procedura
  RET – extrage adresa de revenire din stiva si executa salt la aceasta adresa

Exemple

- C

```
char a, b;
...
a = b;
```

- AVR ASM

```
lds r24, b
sts a, r24
```

Exemple

- C

```
char a;
...
a = 0x10;
```

- AVR ASM

```
ldi     r24, 0x10   ; Load imm 10
sts     a, r24      ; Store to a
```

Exemple

- C

```
int a = *pInt;
```

- AVR ASM

```
; Use the Z register (R31:R30)
lds R30, pInt      ; Load from pInt
lds R31, pInt+1    ;
ld  r24, Z         ; load from (*pInt)
ldd r25, Z+1       ;
sts a,   r24       ; store to a
sts a+1, r25       ;
```

Exemple

- C

```
void strcpy (char *dst, char *src)
{
    char ch;

    do {
        ch = *src++;
        *dst++ = ch;
    } while (ch);
}
```

MOVW	Rd, Rr	Copy Register Word	Rd+1:Rd ← Rr+1:Rr

- AVR ASM

```
; dst in R25:R24, src in R23:R22
strcpy:
    movw r30, r24     ; Z<=dst
    movw r26, r22     ; X<=src
loop:
    ld   r20, X+      ; ch=*src++
    st   Z+, r20      ; *dst++=ch
    tst  r20          ; ch==0?
    brne loop         ; loop if not
    ret
```

Exemple

- C

```
int a, b;
…
a = a + b;
```

- AVR ASM

```
lds   r18, a     ; load a
lds   r19, a+1   ;
lds   r24, b     ; load b
lds   r25, b+1   ;
add   r24, r18   ; add lower half
adc   r25, r19   ; add higher half
sts   a+1, r25   ; store a.byte1
sts   a, r24     ; store a.byte0
```

Exemple

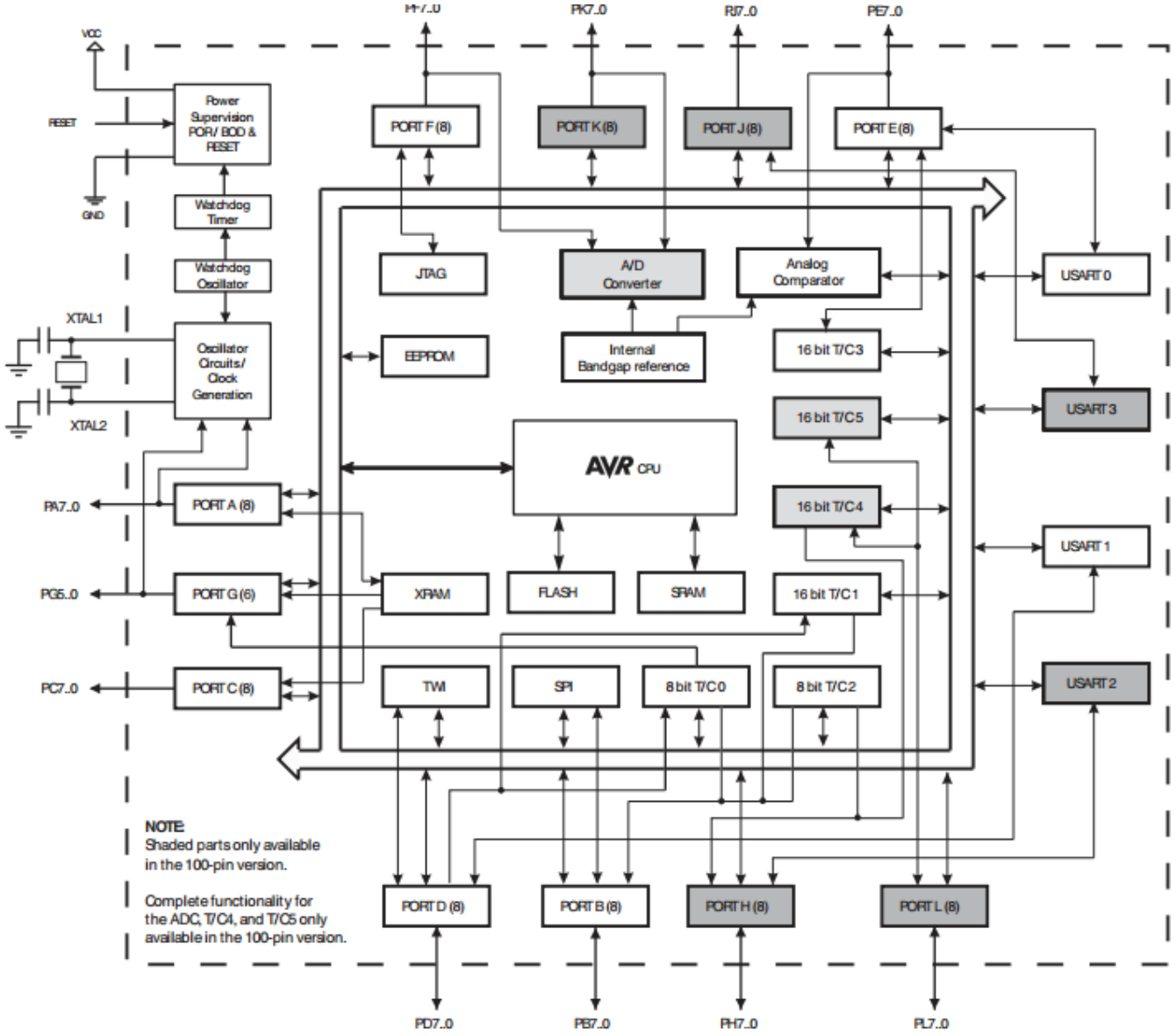
- C

```
char sum, n;
...
for (n = 0; n < 10;
   n++)
   sum += n;
```

- AVR ASM

```
; assume r16=n, r3=sum
        clr     r16       ; n = 0
        rjmp    check
loop:   add     r3, r16   ; sum+= n
        inc     r16       ; n++
check:  cmpi    r16, 10   ; comp n and 10
        brlt    loop      ; br if n<10
```

Microcontrolerul AVR Atmega 2560



Date tehnice Atmega 2560

- 135 Instructiuni, majoritatea executate pe 1 ciclu
- 32 registri pe 8 biti
- Memorie program Flash reprogramabila 256 K Bytes
- Memorie EEPROM 4K Bytes
- Memorie SRAM interna 8K Bytes
- Cicluri de citire/scriere posibile: 10,000 Flash/100,000 EEPROM
- Pana la 64 KB spatii de adresa pentru memorie externa

Periferice integrate pe chip

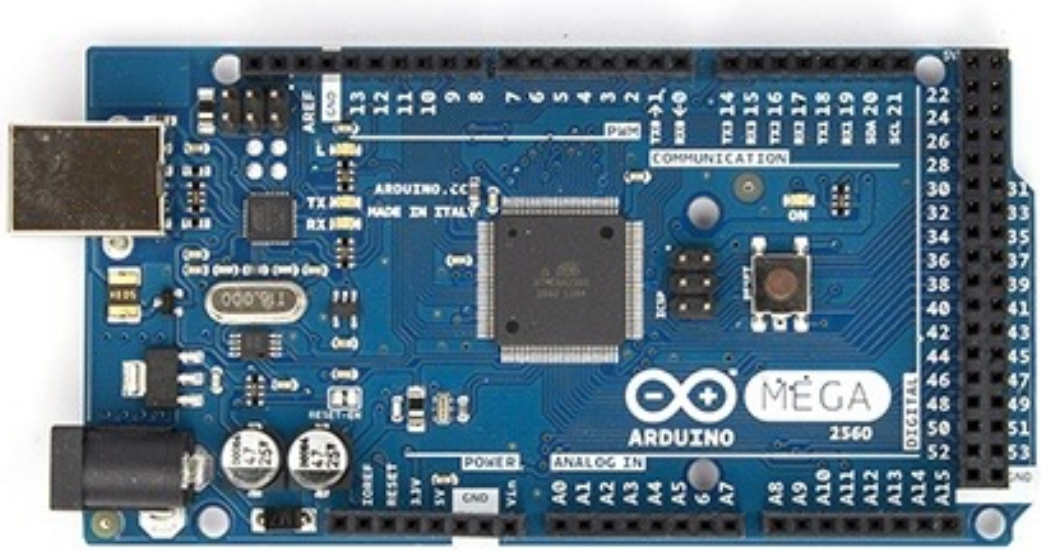
- Doua temporizatoare/numaratoare pe 8-biti
- Patru temporizatoare/numaratoare pe 16 biti
- 4 canale PWM pe 8 biti, 12 canale PWM pe 16 biti
- 16 canale de conversie Analog/Digital pe 10 biti
- 4 interfete programabile USART
- Interfata SPI
- Interfata two-wire (TWI), similara cu I2C
- Generare de intreruperi prin schimbarea starii pinilor

Arduino

- Placi cu microcontroller, si unelte de dezvoltare software open source
- Ascunde detaliile specifice diferitelor microcontrollere, folosind o abordare unificata
- Este disponibila o larga multime de placi, shield-uri si accesorii
- O cantitate impresionanta de documentatie gratuita sau contra cost
- O cantitate impresionanta de exemple pentru orice problema

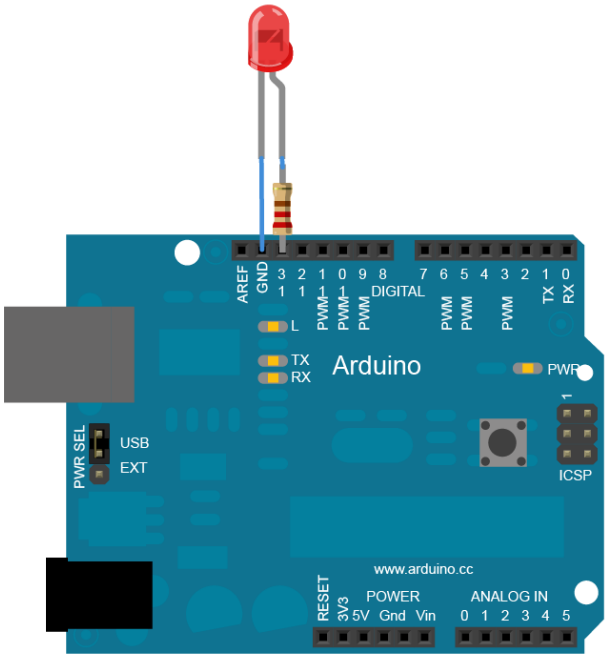
Site web: **www.arduino.cc**
Distribuitori in Romania: **www.robofun.ro**

Arduino Mega 2560



- Bazata pe microcontrollerul ATMega2560, pe 8 biti
- 54 pini de I/O digitali
- 16 pini de intrare pentru semnale analogice
- 4 porturi de comunicare seriala UART
- Frecventa procesorului: 16 MHz
- Alimentare si programare prin cablu USB

Un exemplu de program Arduino



- Aprindere intermitenta a unui LED, conectat la un pin digital de iesire (digital output)

Un exemplu de program Arduino

```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  This example code is in the public domain.
 */

// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);               // wait for a second
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);               // wait for a second
}
```