



Proiectarea cu Micro-Procesoare

Lector: Mihai Negru

An 3 – Calculatoare și Tehnologia Informației

Seria B

Curs 1: Introducere

<http://users.utcluj.ro/~negrum/>



- **Obiective**
 - Cunoașterea, înțelegerea și utilizarea conceptelor: microprocessor, magistrală, memorie, sistem de intrare-ieșire, metode de transfer a datelor, interfețe.
 - Analiza și proiectarea sistemelor cu microprocessor.
- **Cunoștințe 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) și aplicații
 - Partea 2 – aspecte ale proiectării sistemelor cu microprocesor (exemplificate folosind familia x86)
- **Tematica laboratorului**
 - Lucrări practice folosind plăci Arduino (ATmega2560 (MEGA2560), ATmega328P(UNO)) și multiple module periferice (modules)



- **Slide-urile de curs**, disponibile pe site:
 - <http://users.utcluj.ro/~negrum/src/html/dmp.html>
- **Microcontrolere**
 - G. Grindling, B. Weiss, Introduction to Microcontrollers, Vienna Univ. 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.
- **Familia 8086**
 - Barry B. Brey, The Intel Microprocessors: 8086/8088, 80186,80286, 80386 and 80486. Architecture, Programming, and Interfacing, 4-rd edition, Prentice Hall, 1997
 - S. Nedevschi, L. Todoran, „Microprocesoare”, editura UTC-N, 1995, Biblioteca UTCN
- **Documente suplimentare**
 - Data sheets Atmel, Intel etc.
 - Tutoriale Arduino: <https://www.arduino.cc/en/Tutorial/HomePage>



- **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
```

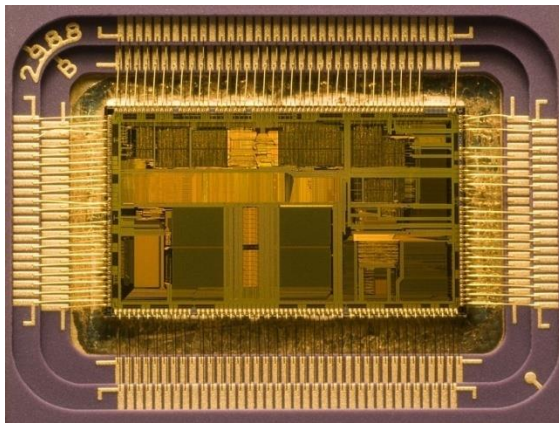
- **Bonus** – se poate acorda pentru activitate deosebită la curs / laborator, sau pentru participarea la concursuri studentești de hardware



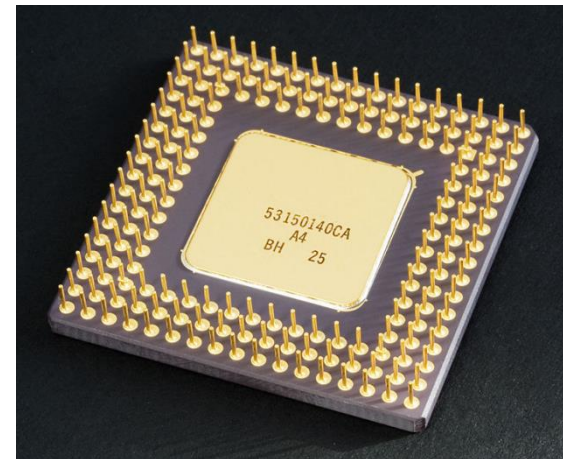
Ce este un microprocessor?



- Un **microprocesor** încorporează toate sau majoritatea funcțiilor unei **unități centrale de procesare** într-un singur circuit integrat.
- O unitate centrala de procesare (**Central Processing Unit, CPU**) este o mașina logică ce poate executa programe de calculator.
- Funcția fundamentală a oricărui CPU, indiferent de forma fizica pe care o are, este sa execute o **secvență de instrucțiuni (programul)**, stocate într-o memorie. Execuția instrucțiunilor se face de obicei în patru pași: citire instrucțiune (**fetch**), decodificare (**decode**), execuție (**execute**) si scriere rezultate (**write back**).



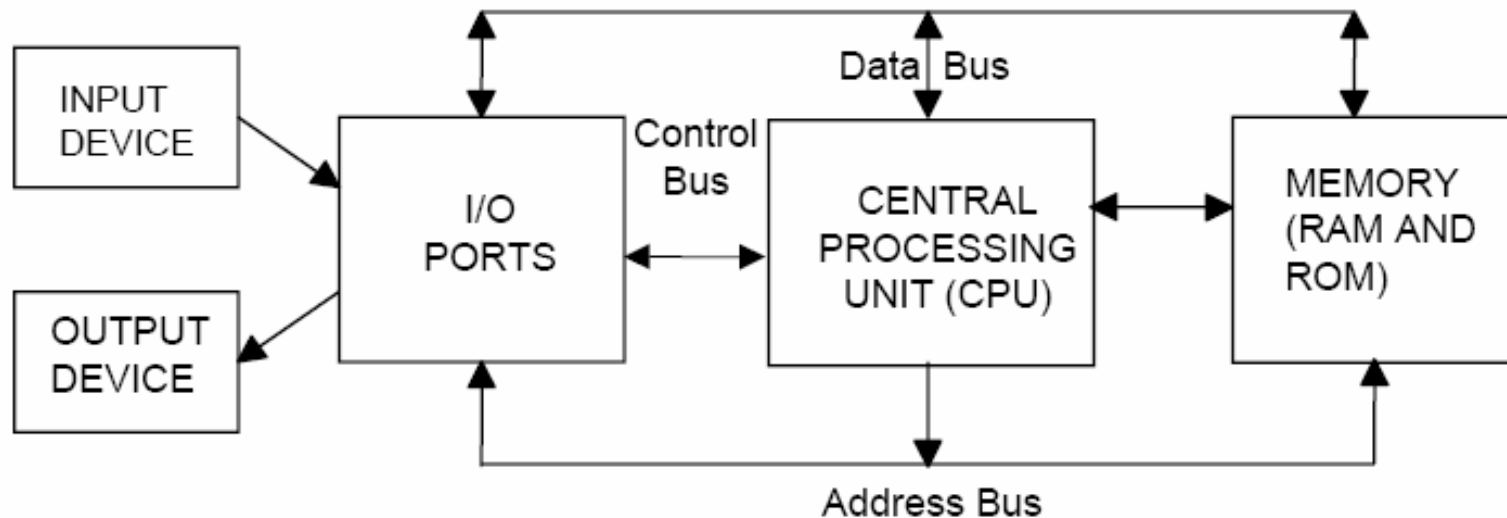
[Intel 80486DX2](#) – interior



[Intel 80486DX2](#) – vedere exteriora



- Microprocesoare
 - **4 bit**: Intel's 4004 (1971), Texas Instruments (TI) *TMS* 1000, si Garrett AiResearch's Central Air Data Computer (CADC).
 - **8 bit**: 8008 (1972), primul procesor pe **8 biți**. A fost urmat de Intel 8080 (1974), Zilog Z80 (1976), si alte procesoare derivate pe 8 biți de la Intel. Competitorul Motorola a lansat Motorola 6800 in August 1974. Arhitectura acestuia a fost clonată si îmbunătățita 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

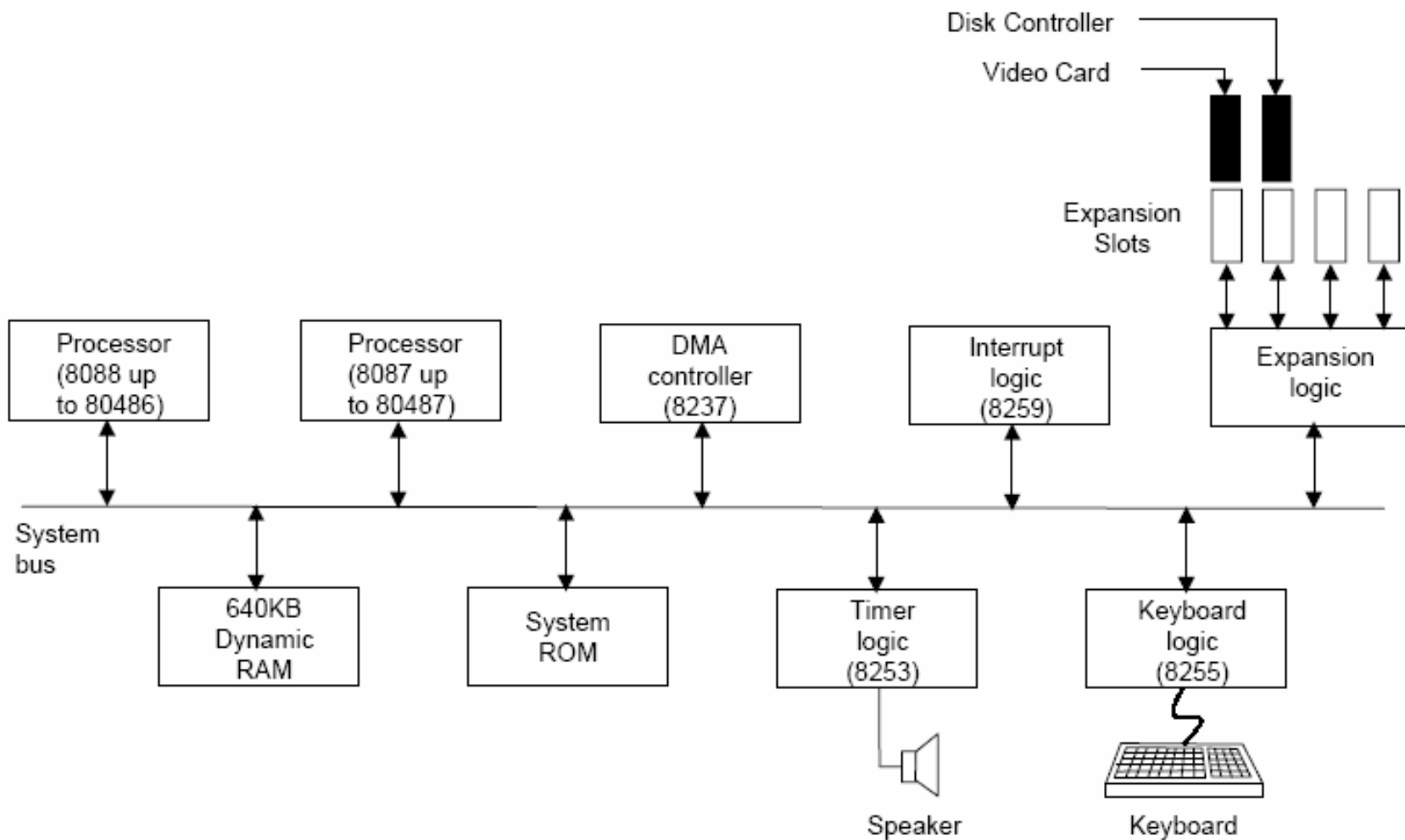


Dispozitive esențiale: CPU, Memorie, I/O

Dispozitive adiționale: Controller întreruperi, DMA, coprocesor, etc

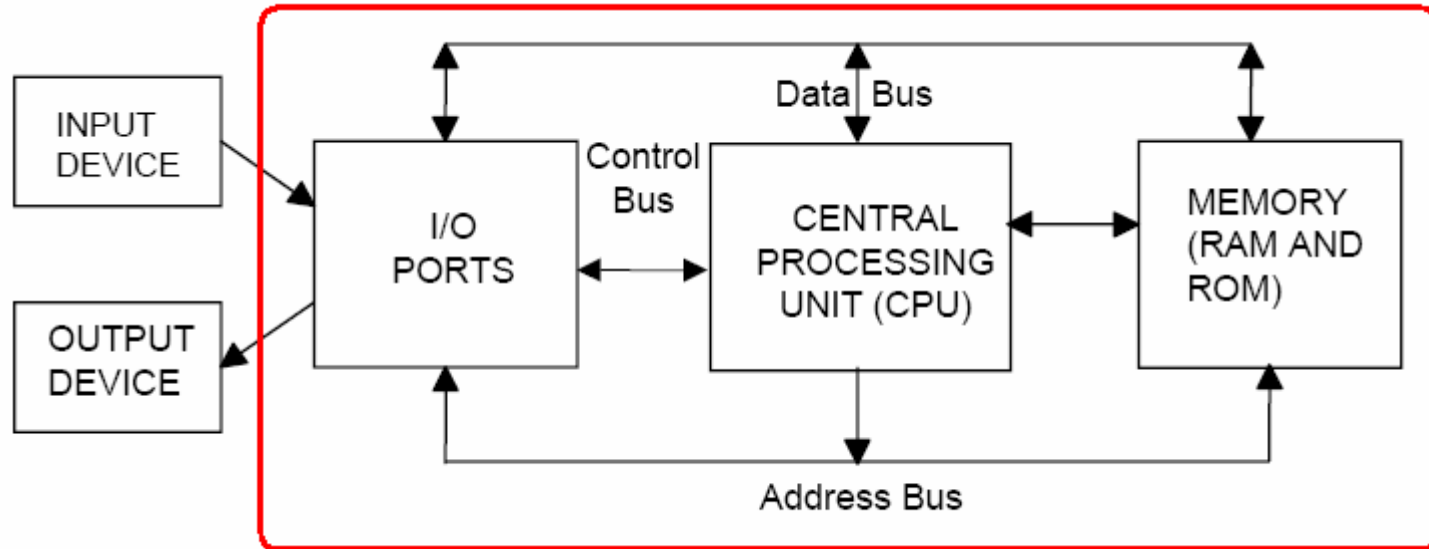


Exemplu: placă de bază PC





Microcontroler (MCU)



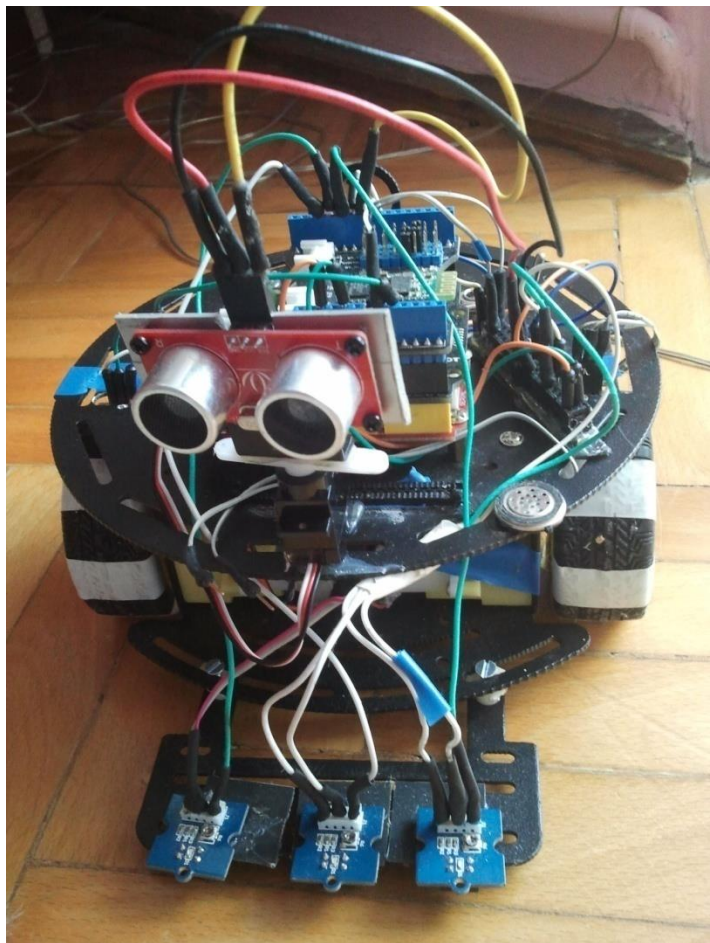
Multiple componente ale unui sistem cu microprocesor sunt incluse in același circuit integrat – Microcontroler

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



- **Obiectiv general:** utilizarea microprocesoarelor (microcontrolerelor) în dezvoltarea de sisteme electronice adaptate unor probleme specifice.
- **Exemple de aplicații:** roboți autonomi, senzori inteligenți, senzori mobili, procesare de semnal audio, procesare de imagini, controlul automat al unor procese, etc.
- **Pașii pentru îndeplinirea acestui obiectiv:**
 - Studiul capabilităților microcontrolerului, familiarizarea cu programarea acestuia
 - Studiul resurselor integrate in microcontroler si resurselor disponibile pe placa cu microcontroler
 - Studiul dispozitivelor externe necesare pentru rezolvarea unor probleme specifice
 - Studiul interfețelor de comunicare, a formatului datelor, si a diagramelor de timp, necesare pentru conectarea microcontrolerului la dispozitivele externe.

- **Exemplu:** proiectarea unui robot capabil să se deplaseze autonom, evitând obstacolele, sau sub controlul unui operator uman, sau ghidat de o bandă de culoare închisă.



Microcontroler: AVR ATmega328, placă Arduino, programare în C/C++

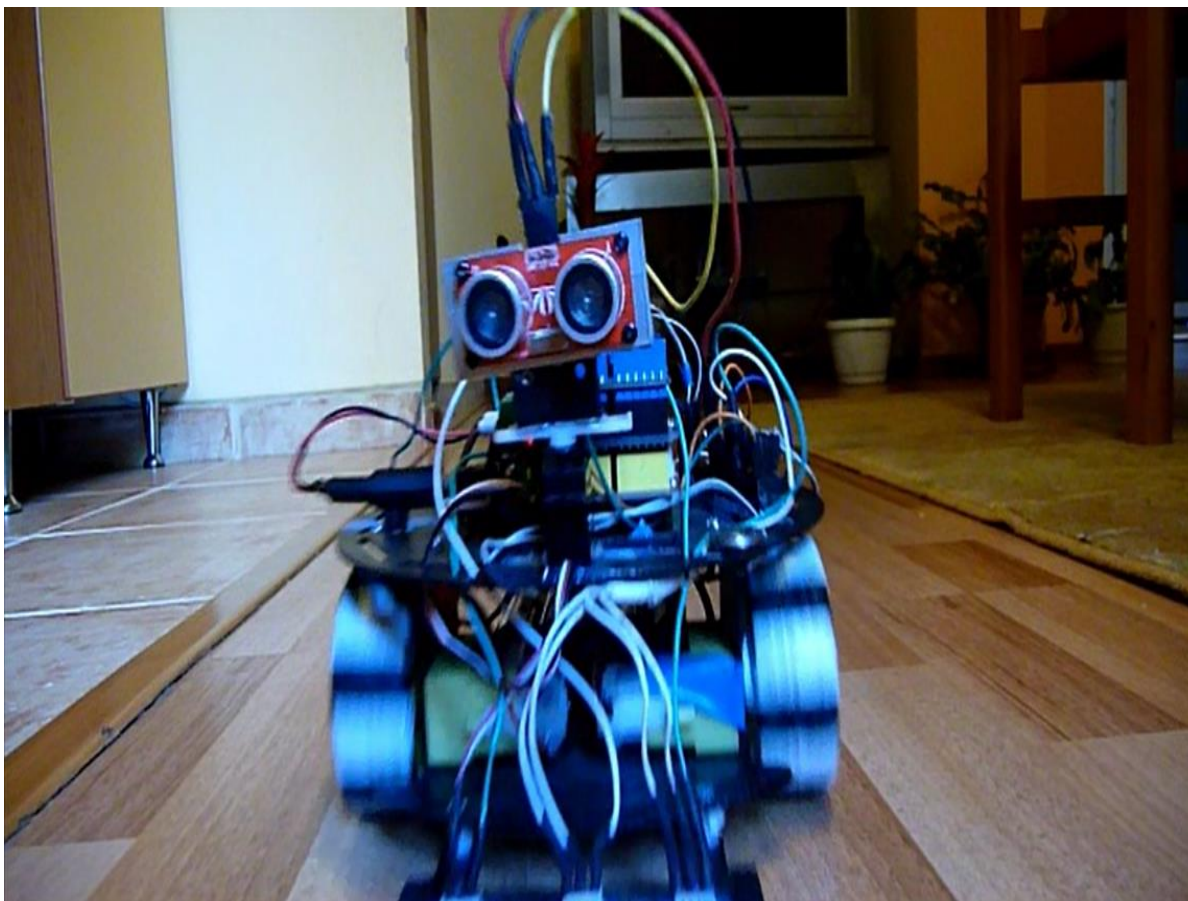
Componente interne: porturi I/O, întreruperi, interfață de comunicare serială, generator PWM

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

Comunicare: serială, tip UART, între MCU și modulul Bluetooth, PWM între MCU și servo, și între MCU și puntea H, semnal analogic de la senzorii de reflectivitate, puls digital între sonar și MCU.

Algoritmi: scanare mediu și detecția obstacolelor, urmărirea liniei, controlul roților pentru mersul în linie dreaptă, etc.

- **Exemplu:** proiectarea unui robot capabil să se deplaseze autonom, evitând obstacolele, sau sub controlul unui operator uman, sau ghidat de o bandă de culoare închisă.



- **Exemplu:** proiectarea unui robot capabil să se deplaseze autonom, evitând obstacolele, sau sub controlul unui operator uman, sau ghidat de o bandă de culoare închisă.

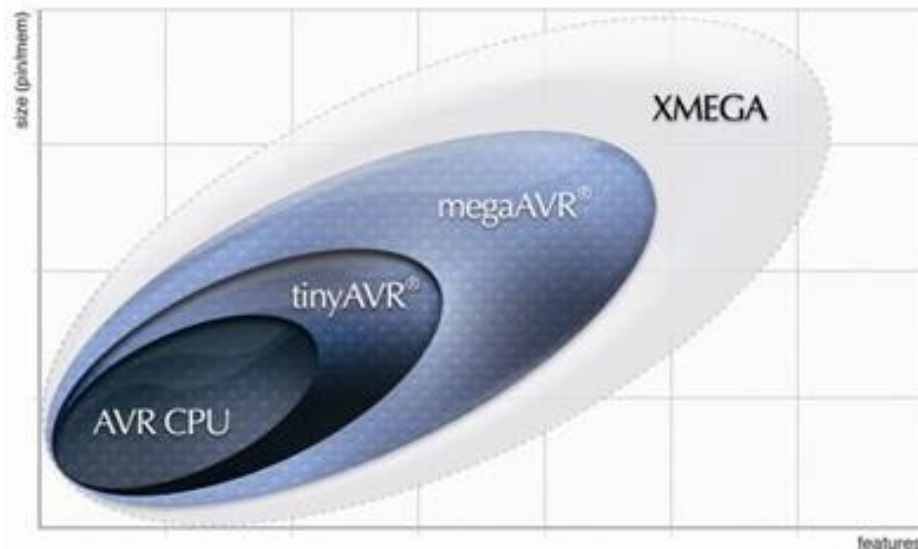




Familia de microcontrolere Atmel AVR 8 biți



- Arhitectura **RISC**
- Execuție 1 instrucțiune / ciclu
- 32 regiștri de uz general
- Arhitectura Harvard
- Tensiune de alimentare 1.8 - 5.5V
- Frecvență controlată software
- Mare densitate a codului
- Gama largă de dispozitive
- Număr de pini variat
- Compatibilitatea integrală a codului
- Familii compatibile între pini și capabilități
- Un singur set de unelte de dezvoltare



tinyAVR

1–8 kB memorie program

megaAVR

4–256 kB memorie program

Set extins de instrucțiuni (înmulțire)

XMEGA

16–384 kB memorie program

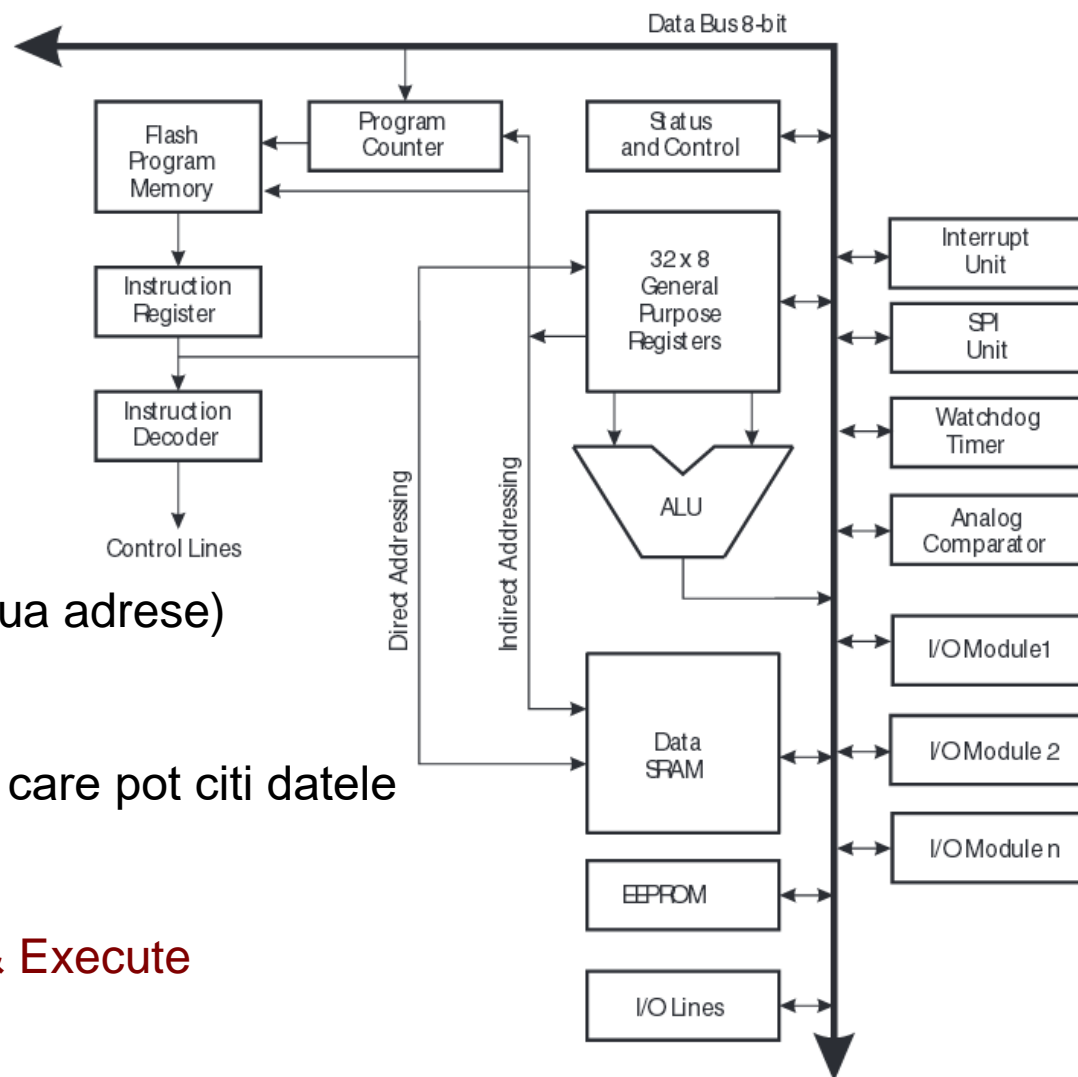
Extra: DMA, suport pentru criptografie

AVR specific pentru aplicații

megaAVR cu interfețe particulare: LCD, USB, CAN etc.



Arhitectura generală a unui microcontroler AVR



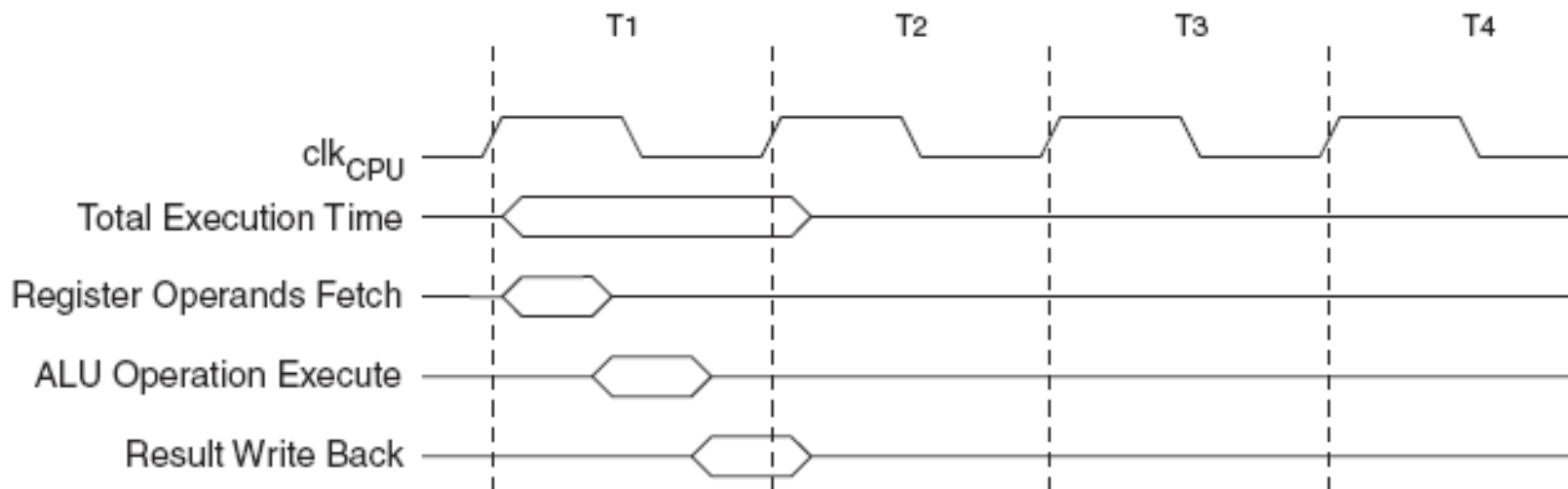
- Mașină RISC (Load-Store cu doua adrese)
- Arhitectura Harvard modificată
 - exista instrucțiuni speciale care pot citi datele din memoria program
- Pipeline pe doua nivele: Fetch & Execute



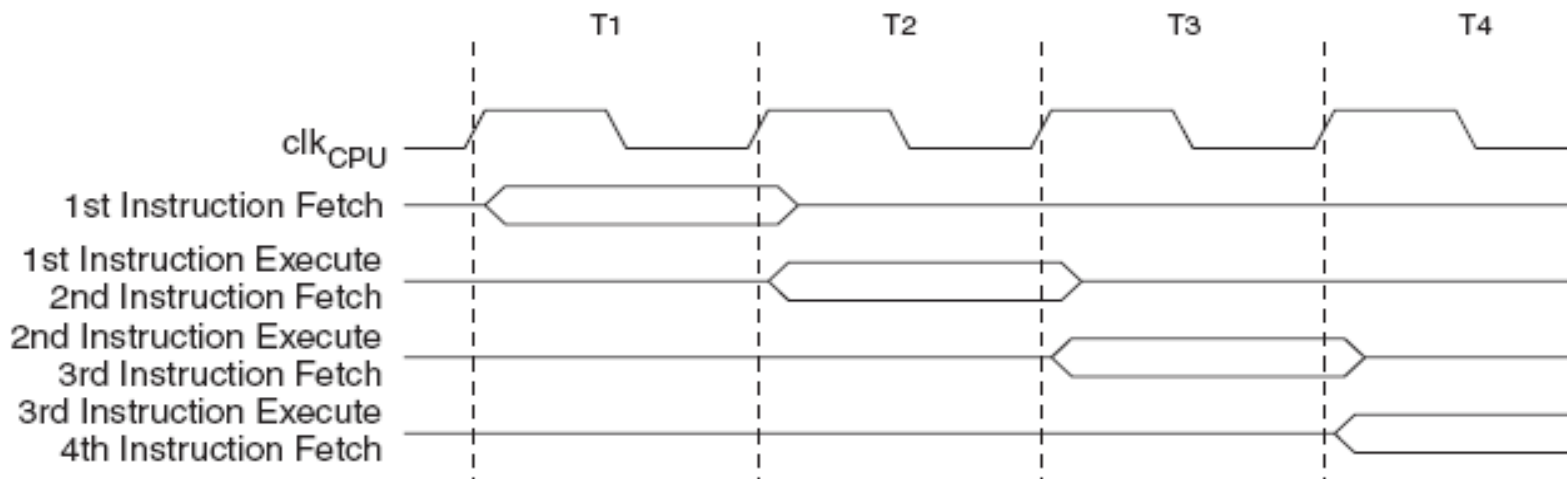
Diagrame de timp AVR



Execuția instrucțiunilor aritmetico-logice



Pipeline asigură suprapunerea citirii următoarei instrucțiuni cu execuția celei curente

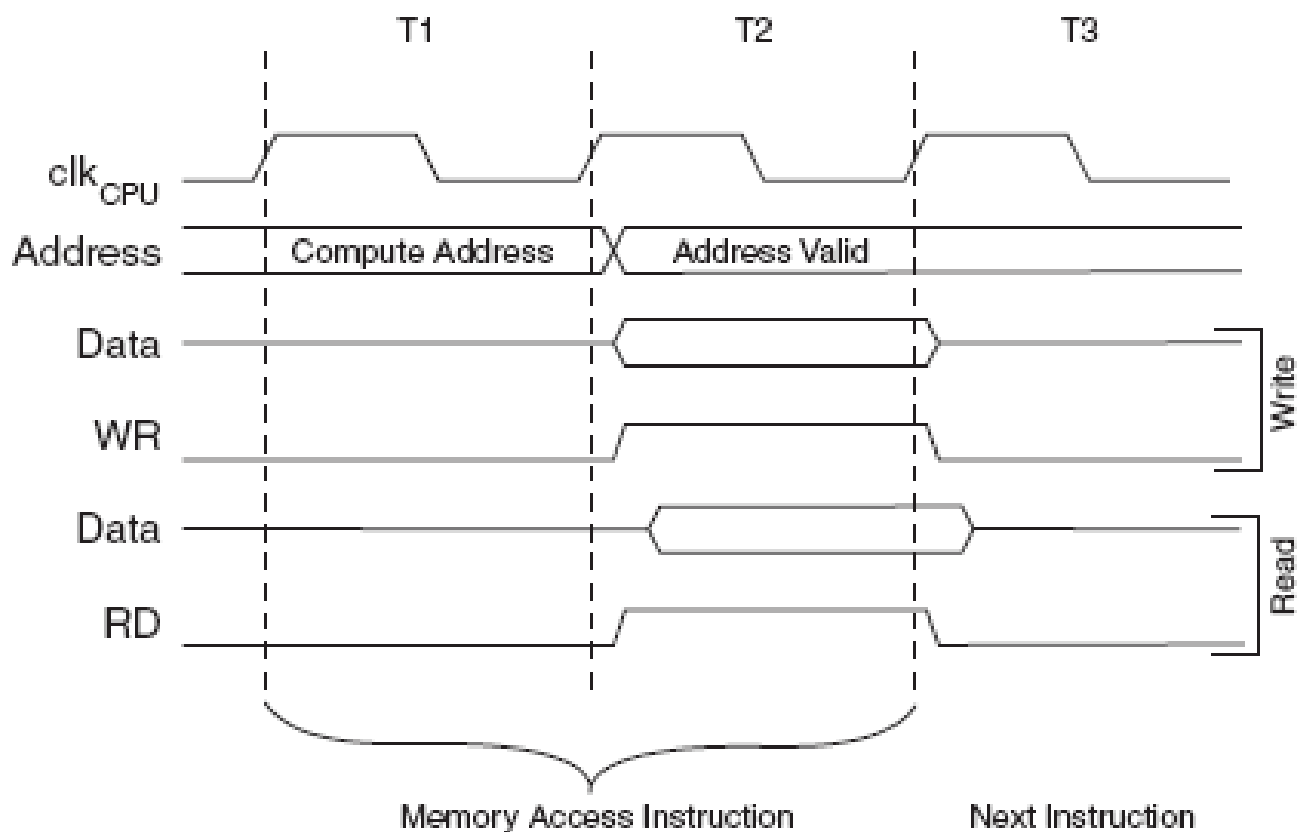




Diagrame de timp AVR



Instrucțiunile care accesează memoria internă SRAM
2 cicluri de ceas / instrucțiune





Registri de uz general (General Purpose Registers – GPR)



- Valori imediate se pot încarcă doar în regiștrii R16-R31
- Regiștrii R26 – R31 sunt folosiți în perechi ca și pointeri
- Fiecare registru are și o adresă în spațiul memoriei de date – adresare uniformă

	7	0	Addr.	
General Purpose Working Registers		R0	0x00	
		R1	0x01	
		R2	0x02	
		...		
		R13	0x0D	
		R14	0x0E	
		R15	0x0F	
		R16	0x10	
		R17	0x11	
		...		
		R26	0x1A	X-register Low Byte
		R27	0x1B	X-register High Byte
		R28	0x1C	Y-register Low Byte
		R29	0x1D	Y-register High Byte
		R30	0x1E	Z-register Low Byte
		R31	0x1F	Z-register High Byte



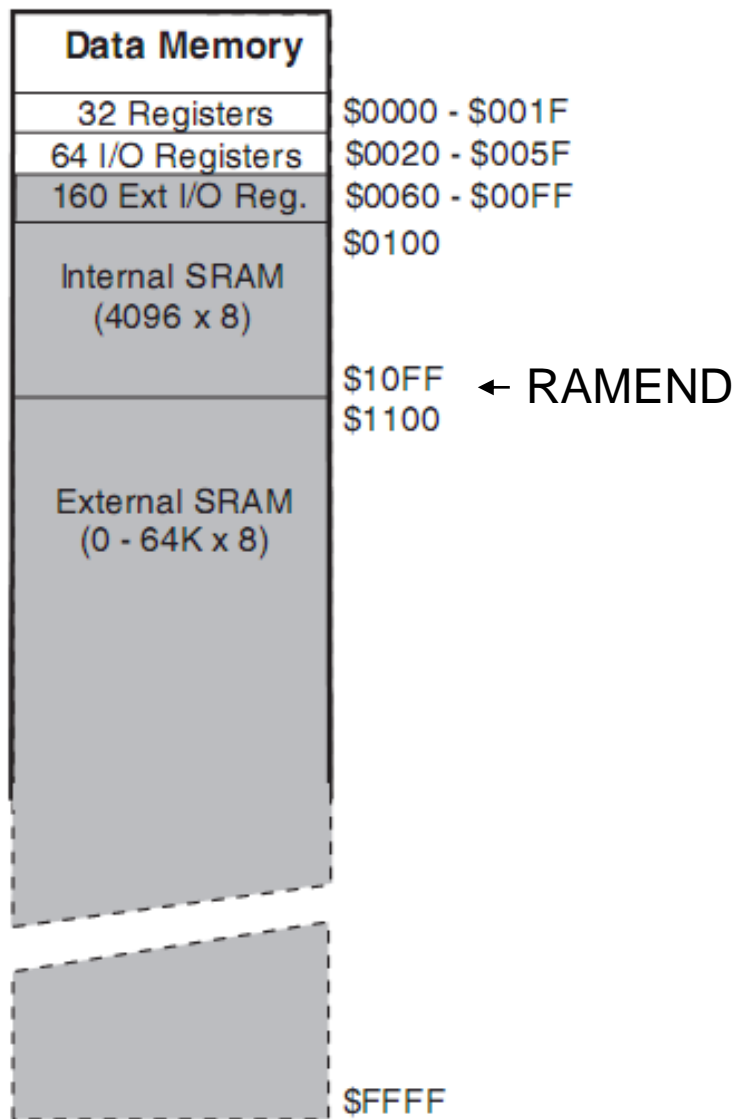
- Copiere date
`mov r4, r7`
- Lucrul cu valori imediate – posibil doar cu regiștrii r16 – r31
`ldi r16, 5`
`ori r16, 0xF0`
`andi r16, 0x80`
`subi r20, 1`
- Operații aritmetice și logice între regiștri
`add r1, r2`
`or r3, r4`
`lsl r5`
`mul r5, r18` – $r1:r0 = r5 * r18$
`rol r7`
`ror r9`
`inc r19`
`dec r17`



Memoria de date



- Primele 32 de adrese – blocul de regiștri
- 64 de adrese – regiștri I/O accesabili prin instrucțiuni speciale
- 160 adrese – spațiu I/O extins, accesabil prin instrucțiuni standard de acces la memorie
- SRAM, de ordinul Kbytes (2, 4, 8 ...)
- Posibilitate de extensie pana la 64 KB
- Constanta predefinita RAMEND marchează sfârșitul memoriei de date interne





Operații cu memoria de date



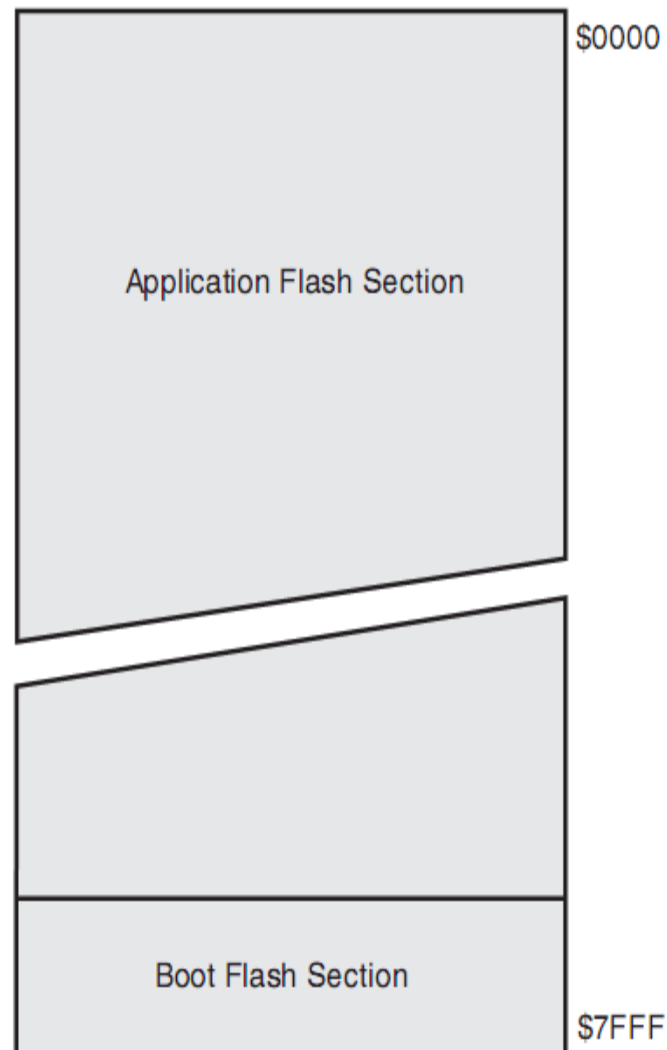
- Accesarea directă a memoriei de date
 - lds r3, 0x10FE**
 - lsl r3**
 - sts 0x10FE, r3**
- Accesarea indirectă a memoriei de date, prin intermediul regiștrilor X, Y, Z
 - ldi r27, 0x10** octetul superior al lui X este r27
 - ldi r26, 0xFE** octetul inferior al lui X este r26
 - ld r0, X**
 - lsl r0**
 - st X, r0**
- Accesarea cu autoincrementare/decrementare a adresei
 - ld r0, X+** se accesează locația X, apoi se incrementează X
 - ld r0, +X** se incrementează X, apoi se accesează locația X
 - ld r0, X-**
 - ld r0, -X**



Memoria program



- Memorie flash, pentru programarea aplicațiilor
- Organizată în cuvinte de 16 biți
- Două secțiuni: Boot și Aplicație
- Cel puțin 10.000 cicluri scriere/ștergere
- Constante pot fi declarate în segmentul de cod, ele vor fi stocate în 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 destinație, Z adresa
 - **Scrierea** – doar pe cuvânt
 - SPM r0 este destinație, Z adresa
 - PM(Z) <= R1:R0





Registrul de stare SREG



- Registrul SREG (8 biți) conține informații despre starea sistemului și rezultatul unor operații
- Folosit pentru modificarea comportamentului programului sau pentru salturi condiționate
- **Nu este salvat automat la apelul procedurilor sau la execuția întreruperilor!**

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 întreruperilor
- T – bit de transfer, poate fi copiat prin instrucțiunile BLD și BST din alt registru
- H – carry între jumătăți de octet, folositor pentru operații BCD
- S – $N \text{ xor } V$, pentru teste între numere cu semn
- V – indicator de overflow la operații în complement fata de 2
- N – indicator de rezultat negativ
- Z – indicator al unui rezultat nul
- C – carry



- **Salturi necondiționate**

 - RJMP** – salt relativ, PC +- 2KB

 - JMP** – salt absolut

 - IJMP** – salt indirect, la adresa indicata de pointerul Z

- **Salturi condiționate**

 - CP, CPI** – compară 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 următoarea instrucțiune daca un bit într-un registru e setat

 - SBRS r5, 2 – daca bitul 2 din reg. 5 este setat, execută saltul

 - SBRC, SBIS, SBIC** (I/O register set/clear)

- **Apeluri de procedură**

 - RCALL, CALL, ICALL** – se salvează adresa de revenire în stiva, nu se salvează nimic altceva

- **Revenire din procedură**

 - RET** – extrage adresa de revenire din stivă și execută salt la această adresă



Exemple

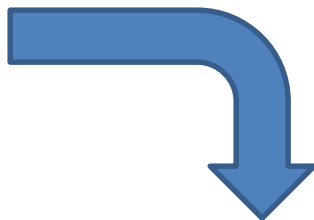


C
char a, b;
...
a = b;



AVR ASM
lds r24, b
sts a, r24

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);
}
```

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
```

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



Exemple



C

AVR ASM

```
int a, b;
```

```
lds    r18, a    ; load a
```

```
lds    r19, a+1  ;
```

```
...
```

```
lds    r24, b    ; load b
```

```
a = a + 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

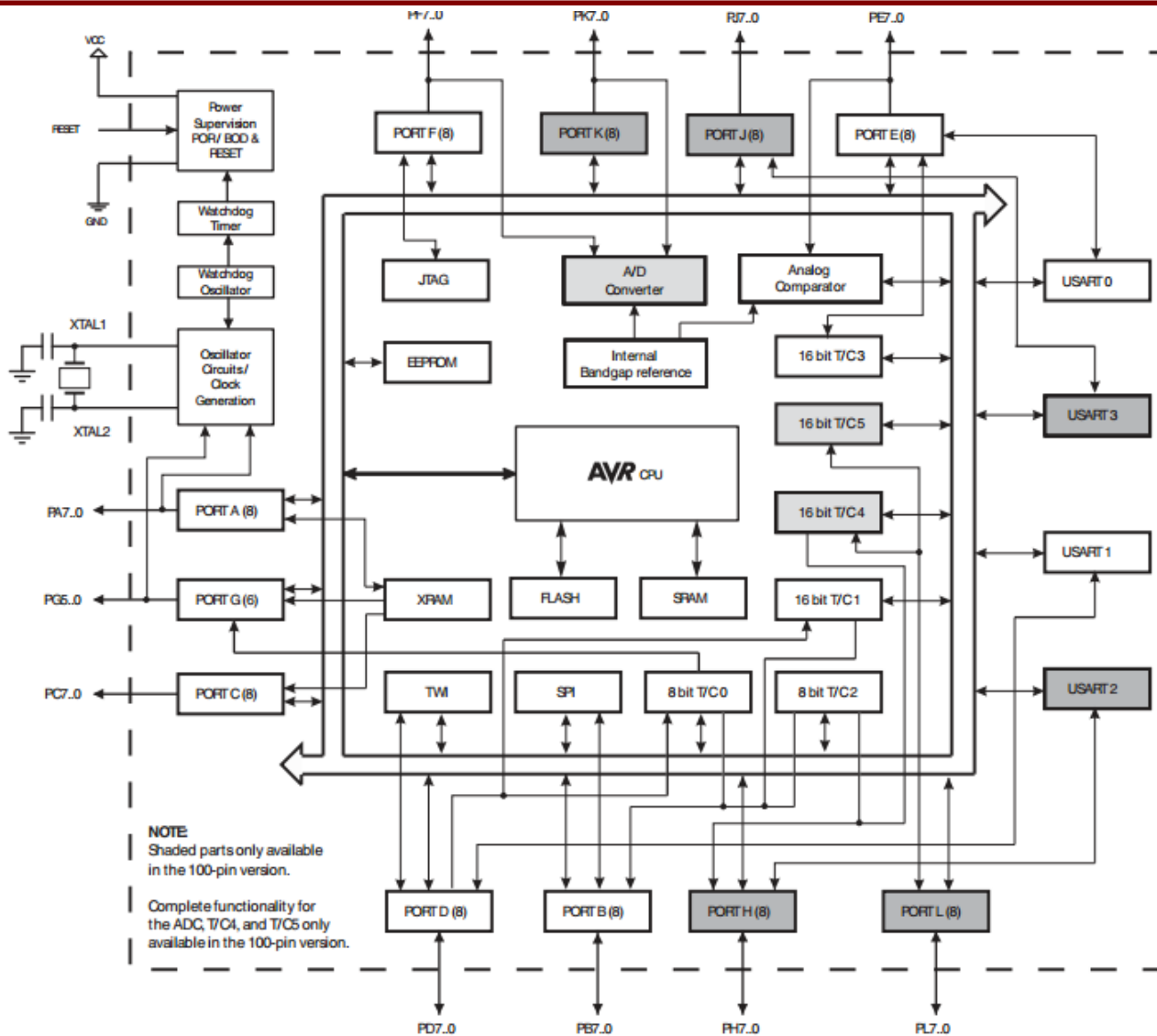
```
char sum, n;  
...  
for (n = 0; n < 10;  
    n++)  
    sum += n;
```

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 Instrucțiuni, majoritatea executate pe 1 ciclu de ceas
- 32 regiștri pe 8 biți
- Memorie program Flash reprogramabilă – 256 K Bytes
- Memorie EEPROM – 4 K Bytes
- Memorie SRAM internă – 8 K Bytes
- Cicluri de citire/scriere posibile: 10,000 Flash / 100,000 EEPROM
- Pană la 64 KB spații de adresă pentru memoria externă

- **Periferice integrate pe chip**
- Două temporizatoare / numărătoare pe 8-biti
- Patru temporizatoare / numărătoare pe 16 biți
- 4 canale PWM pe 8 biți, 12 canale PWM pe 16 biți
- 16 canale de conversie Analog / Digital pe 10 biți
- 4 interfețe programabile USART
- Interfață SPI
- Interfață two-wire (TWI), similară cu I2C
- Generare de întreruperi prin schimbarea stării pinilor



Arduino



- Plăci cu microcontroler, și unelte de dezvoltare software open source
- Ascunde detaliile specifice diferitelor microcontrolere, folosind o abordare unificată
- Este disponibilă o largă mulțime de placi, shield-uri și accesorii
- O cantitate impresionantă de documentație gratuită sau contra cost
- O cantitate impresionantă de exemple pentru orice problemă

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



Arduino Mega 2560



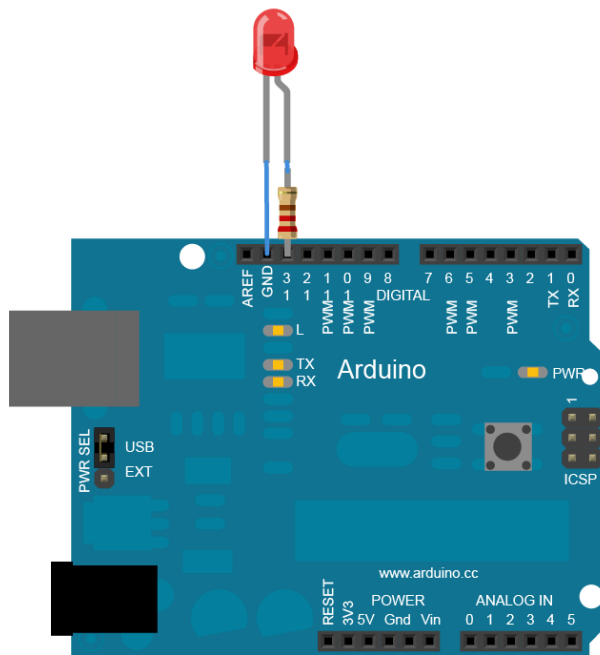
- Bazată pe microcontrolerul ATmega2560, pe 8 biți
- 54 pini de I/O digitali
- 16 pini de intrare pentru semnale analogice
- 4 porturi de comunicare serială UART
- Frecvența procesorului: 16 MHz
- Alimentare și programare prin cablu USB



Un exemplu de program Arduino



- Aprindere intermitentă a unui LED, conectat la un pin digital de ieșire (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  
}
```