ATmega103 Assembly II

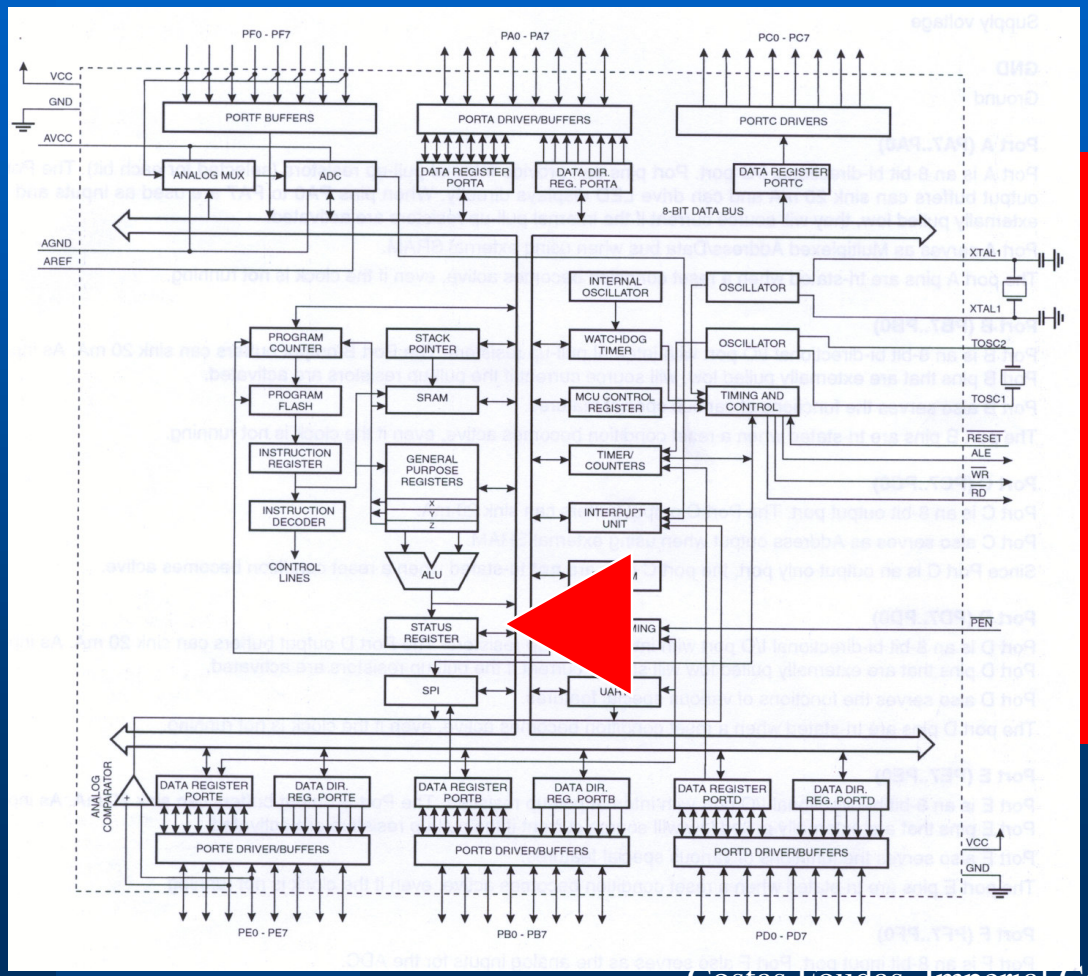
Costas Foudas, Imperial College,
Rm: 508, x47590

Outline:

- The ATmega103 Status Register
- Pointer Registers
- Branch instructions
- Defining Tables in the Program memory
- Subroutines
- Elementary example program

Costas Foudas, Imperial College,
Rm: 508, x47590

The ATmega103 Status register



- **SREG monitors the ALU activity**
- **SREG is used by branch, compare and arithmetic instructions**

Status Register bit definitions:

D7	D6	D5	D4	D3	D2	D1	D0
I	T	H	S	V	N	Z	C
♠	♠	♠	S=N⊕V	2's Complement Overflow	NEG.	ZERO	CARRY

- **SREG simply just another Port in the I/O region and can be accessed via IN/OUT commands**

♠ The higher bits are subject of more advanced discussions reserved for later.

Costas Foudas, Imperial College,
Rm: 508, x47590

Exercising the Status register

- Write a little program to read the Status Register:

```
Main:
        LDI  r16, $55
        CLR  r17
        CPI  r16, $54
        IN  r17, SREG
        OUT PORTB, r17
        rjmp Main
```

Change this number and see how the SREG bits change

CPI Rd, K : Rd – K
Flags: Z,N,V,C,H

Exercising the Status register

Exercise: Try to set the various status register bits using commands (like CPI) that would set them.

Hint: Check the documentation of the ATmega103 assembly commands (in the course web page) and find commands that can set the appropriate flags.

Pointer Registers I

- **The pairs of registers:**

  r31:r30 (Z), r29:r28 (Y), r27:r26 (X)

  are special pointer registers and can be used to access memory via :

  LD Rd, X or LD Rd, Y or LD Rd, Z

  Meaning: Load the contents of memory address X or Y or Z on register Rd

Costas Foudas, Imperial College,
Rm: 508, x47590

Pointer Registers II

- **We can chose to increment memory address after the operation is finished:**

```
LD Rd, X+    LD Rd, Y+   LD Rd, Z+
```

or decrement before the operation :

```
LD Rd, -X     LD Rd, -Y    LD Rd, -Z
```

Pointer Registers III

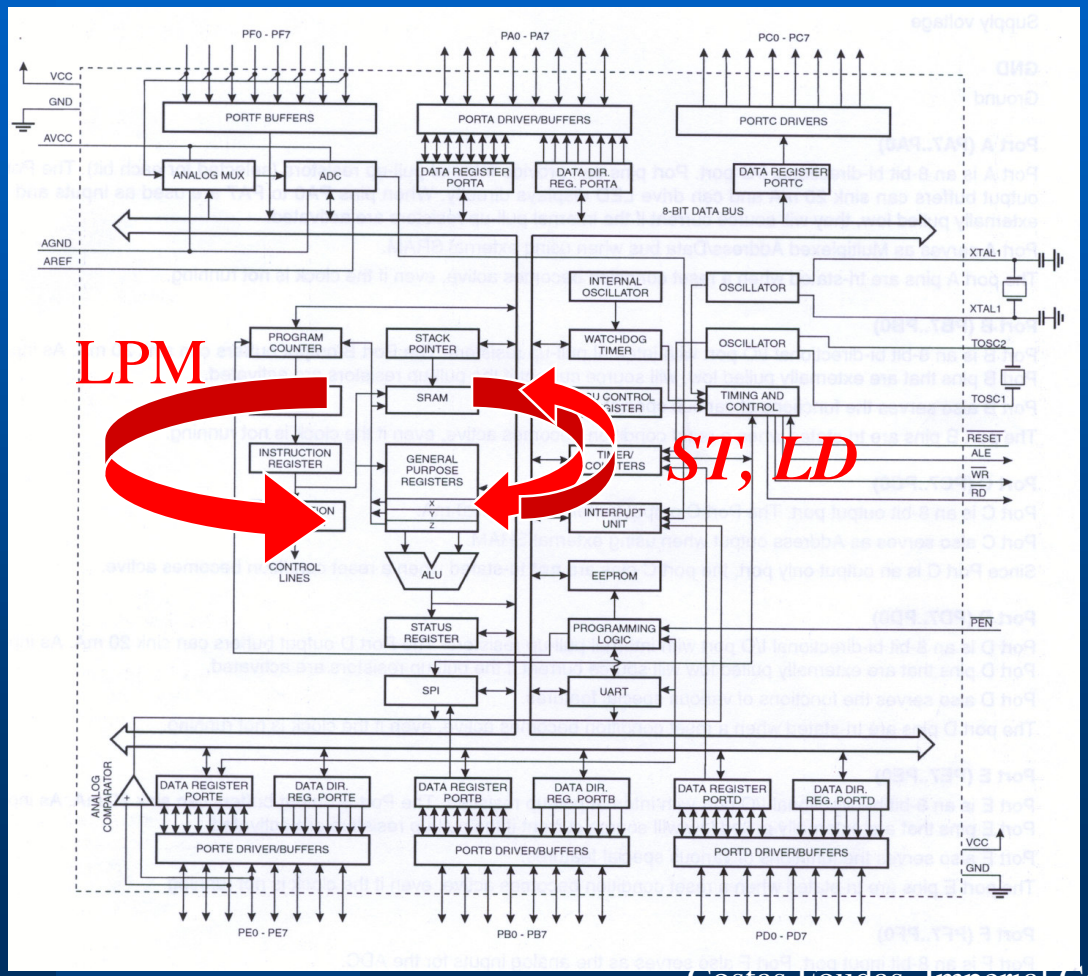
- **To store the register contents to the memory use :**

```
ST  X,  Rx     ST Y, Rx        ST Z, Rx

ST  X+,  Rx    ST Y+, Rx       ST Z+, Rx

ST  -X,  Rx    ST -Y, Rx       ST -Z, Rx
```

Pointer Registers IV



This way data can be moved from the data SRAM to the register file and the reverse using LD and ST commands

Data can be moved from the program flash memory to the register file ONLY via the Z-register and by using the LPM command.

Pointer Registers V

- **Special command which moves data from the program memory at address Z to register r0 :**

```
LPM
```

Subroutines

- **Separate tasks using subroutines:**

Within your main program you can call subroutines by: ***rcall delay0***

```
Delay0:
        INC  r16
        BREQ  finito
        rjmp Delay0
finito: ret
```

Branch Instructions

- **In the previous example:**

```
Delay0:
        INC  r16
        BREQ  finito
        rjmp Delay0
finito: ret
```

The command ***BREQ*** will inspect the SREG
After the ***INC*** command has been executed and
If it is found that the ZERO bit is set it will instruct the
Processor to jump to '***finito***'

Exercises:

Exercise 1 : **Last time you made a counter. Use now the a subroutine to delay the speed of the counter.**

Exercise 2 : **You can delay further by cascading delay subroutines.**

Exercise 3 : **Write a program that writes incrementing numbers to the SRAM. Read them back, compare with what you expect to see and if it is not correct send a pattern of light at PORTB (memory test program). What happens if you do not start writing from address $60 .**

Tables in the Program Memory

- **Convenient way to store a data pattern in the program memory:**

```
MyTable:
.DW $0100, $0302, $0504, $0706, $0908      ;Table loading with 16 bit words
.DW $0B0A, $0D0C, $0F0E, $1110, $1312      ;Next 5 16-bit words in the table
.DW $1514, $1617, $1918, $1B1A, $1D1C      ;Next 5 16-bit words in the table
.DW $1F1E                                  ;Last word in the table
```

Accessing Program Memory I

- **Access the table :**

```
MyTable:
.DW $0100, $0302, $0504, $0706, $0908      ;Table loading with 16 bit words
.DW $0B0A, $0D0C, $0F0E, $1110, $1312      ;Next 5 16-bit words in the table
.DW $1514, $1617, $1918, $1B1A, $1D1C      ;Next 5 16-bit words in the table
.DW $1F1E                                  ;Last word in the table
```

- **Need to find out the address of this table in the microprocessors program memory !**

Accessing Program Memory II

- **To calculate the address use:**

```
Start:
        ldi  ZH, HIGH(MyTable*2)      ; load the Table address to Z (High Byte)
        ldi  ZL, LOW(MyTable*2)       ; load the Table address to Z (Low Byte)
        SUBI ZL, $01
        rjmp EndStart
EndStart:     ret
```


Costas Foudas, Imperial College,
Rm: 508, x47590

Accessing Program Memory III

- **Use then LPM to bring the contains of Z to r0 :**

```
ADIW ZL, 01          ; point to the table address
LPM                  ; bring the low byte to r0
MOV r24, r0          ; move the result to register 24
ADIW ZL, 01          ; now increment the address by one
                     ;to get the high byte
LPM                  ;bring high byte to r0
MOV r25, r0          ; put it in register 25
```

Exercise

- Write a program that reads the contents of a table in the memory and sends the output to PORTB.
- Use subroutines to organize your program.
- Use delays to see the different patterns flashing as they light the LEDS of PORTB.