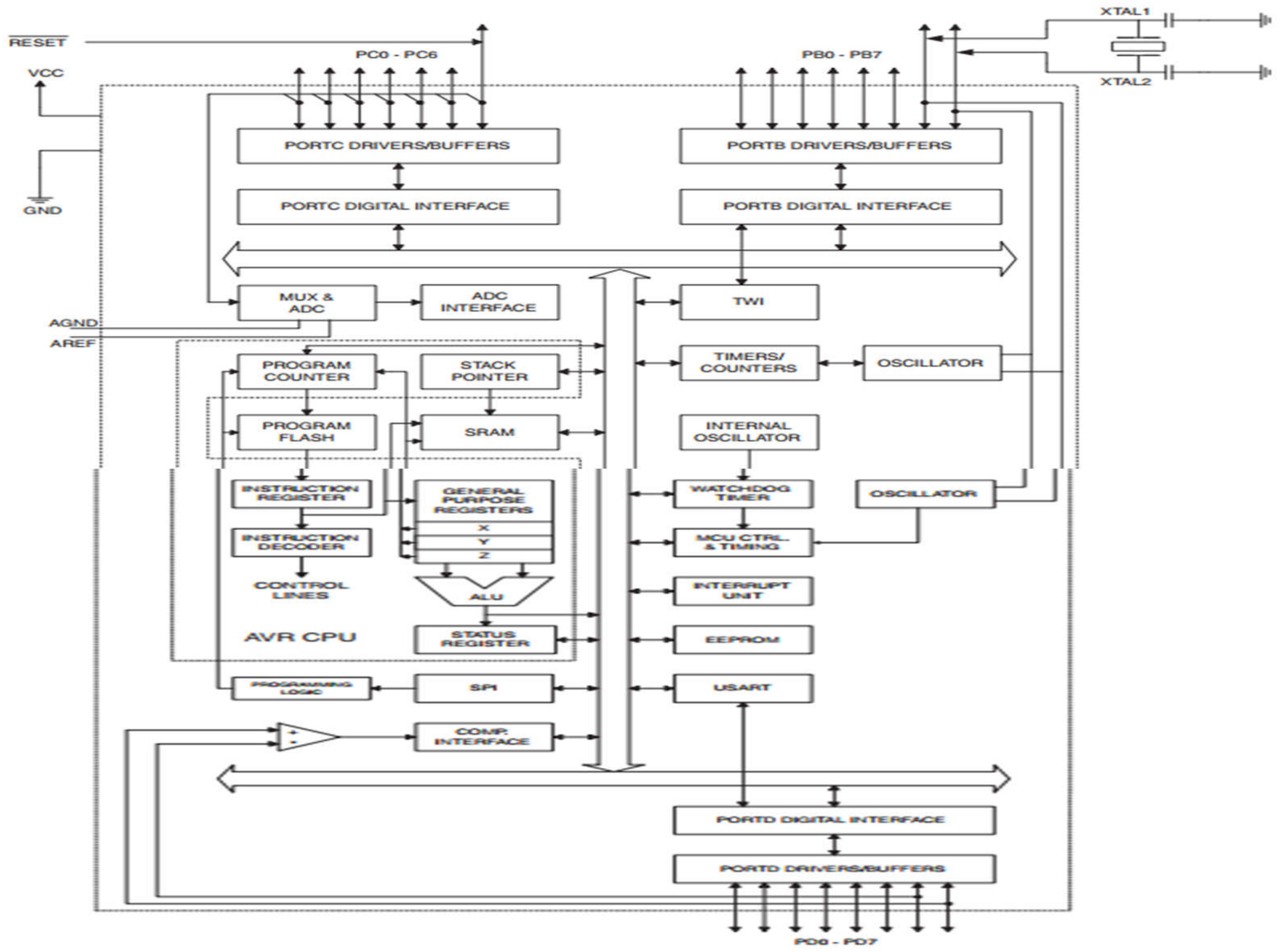
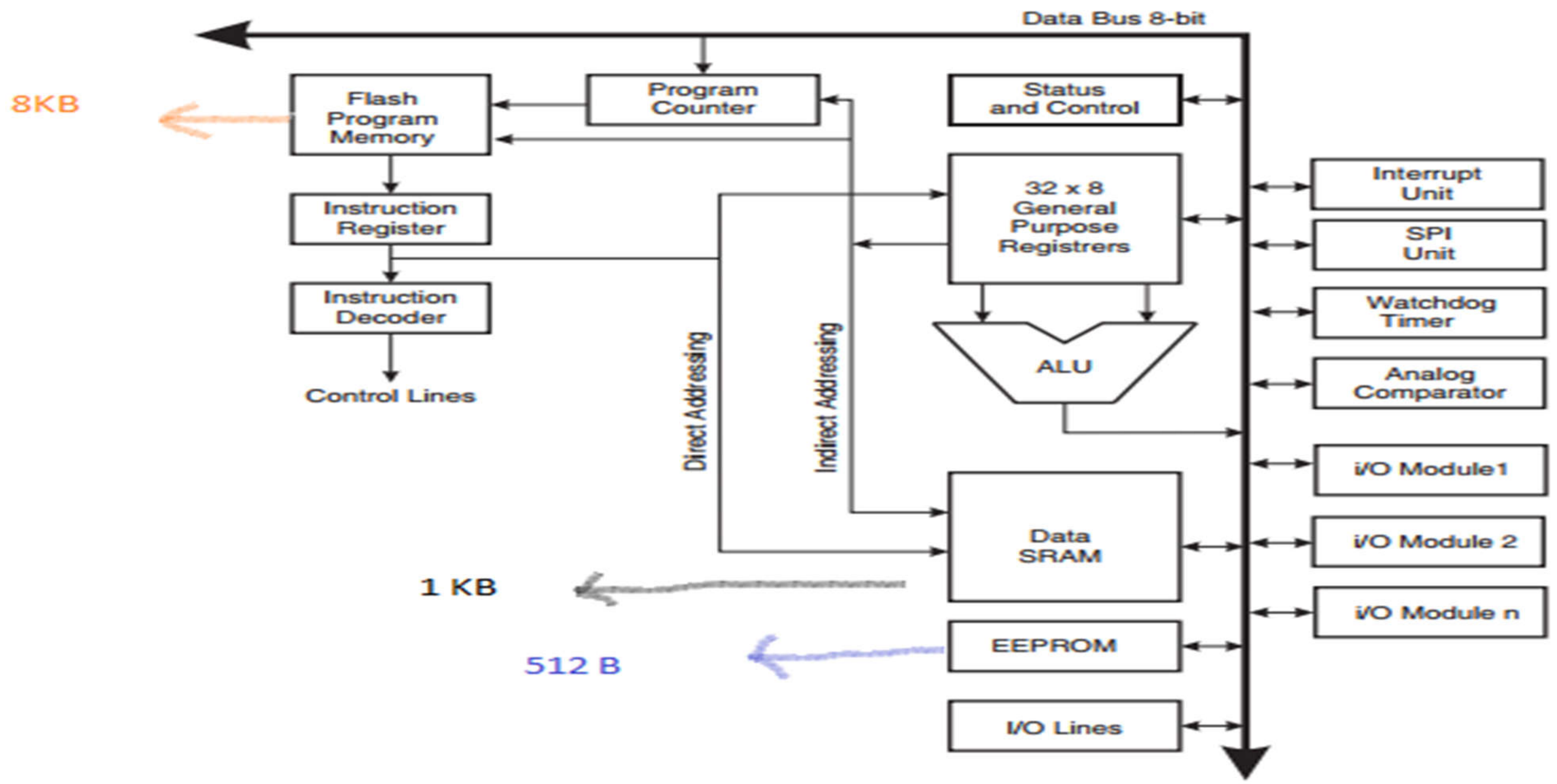
AVR ATmega8 Block Diagram/Internal Architecture



AVR MCU Architecture



Description of different blocks of MCU

- AVR use Harvard Architecture to maximize performance and parallelism : separate memories and buses for program and data.
- Instructions in the Program memory are executed with a single level pipelining :While one instruction is being executed, the next instruction is pre-fetched from the Program memory.
- This concept enables instructions to be executed in every clock cycle.
- Fast access Register File contains 32x8-bit general purpose registers with a single clock cycle access time.
- Single cycle ALU operation.

- **ALU Operation:** Two operands are loaded from registers into two ALU inputs, operation is executed and result output is stored back in register in one clock cycle.

- The ALU supports arithmetic and logic operations between registers or between a constant and a register.

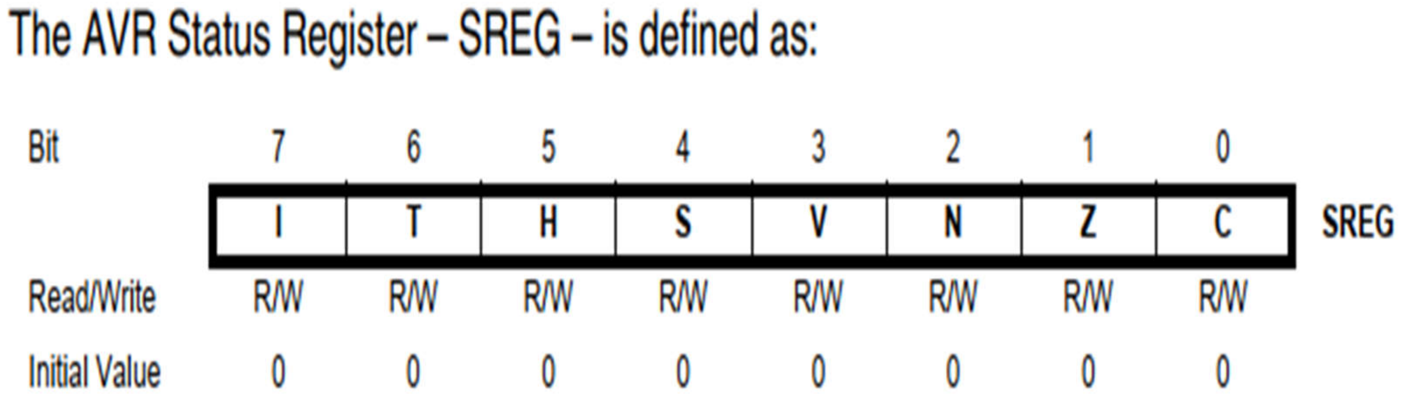
- Single register operations can also be executed in the ALU.

- After an arithmetic operation, the Status Register is updated to reflect information about the result of the operation.

-
  The ALU operations are divided into three main categories – arithmetic, logical, and bit-functions. Multiplier supporting both signed/unsigned multiplication and fractional format

Status Register

- The Status Register contains information about the result of the most recently executed arithmetic instruction.



The AVR Status Register – SREG – is defined as:

Bit	7	6	5	4	3	2	1	0	
	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	

Status bit description

- **Bit 7-(I):Global interrupt enable**

The Global Interrupt Enable bit must be set for the interrupts to be enabled. The I-bit is cleared by hardware after an interrupt has occurred, and is set by the **RETI** (interrupt return) (PC<-stack) instruction to enable subsequent interrupts.

The I-bit can also be set and cleared by the application with the **SEI (global interrupt enable)** and **CLI(global interrupt disable)** instructions

Status Register Description

- **Bit 6 – T: Bit Copy Storage**

The Bit Copy instructions BLD (Bit LoaD) and BST (Bit STore) use the T-bit as source or destination for the operated bit.

A bit from a register in the Register File can be copied into T by the BST instruction.

$$T \longleftarrow Rs(b)$$ (Bit and Bit Test instructions)

A bit in T can be copied into a bit in a register in the Register File by the BLD instruction.

$$Rd(b) \longleftarrow T$$

Status Register Description

- **Bit 5 – H: Half Carry Flag**
  The Half Carry Flag H indicates a Half Carry in some arithmetic operations.

  Half Carry is useful in BCD arithmetic. This Flag is affected by Arithmetic and logic instructions. E.g ADD, ADC instructions etc.

$Rd \longleftarrow Rd + Rs$ (add two registers)

$Rd \longleftarrow Rd + Rs + 1$ (add with carry two Reg.)

Status Register Description

- **Bit 4 – S: Sign Bit, S = N XOR V:**

The S-bit is always an exclusive or between the Negative Flag N and the Two's Complement Overflow Flag V. E.g SES (set signed test flag), CLS(clear signed test flag).

$$S \longleftarrow 1 \quad \text{(SES instruction)}$$

$$S \longleftarrow 0 \quad \text{(CLS instruction)}$$

Bit 3 – V: Two's Complement Overflow Flag :

The Two's Complement Overflow Flag V supports two's complement arithmetics
E.g SEV (set two's complement overflow, CLV (clear two's comp overflow)

$$V \longleftarrow 1 \text{ (SEV)}$$

$$V \longleftarrow 0 \quad \text{(CLV)}$$

Status Register Description

Bit 2 – N: Negative Flag

The Negative Flag N indicates a negative result in an arithmetic or logic operation . E.g SEN, CLN, LSL, LSR, ROL, ROR etc.

N $\longleftarrow$ 1 (SEN) (set negative flag)

N $\longleftarrow$ 0 (CLN)(clear negative flag)

Status Register Description

- **Bit 1 – Z: Zero Flag**
  The Zero Flag Z indicates a zero result in an arithmetic or logic operation.
- E.g: ADD, ADC (addition of two registers with carry), SUB (subtract two registers, SUBI(subtract constant from register) etc.

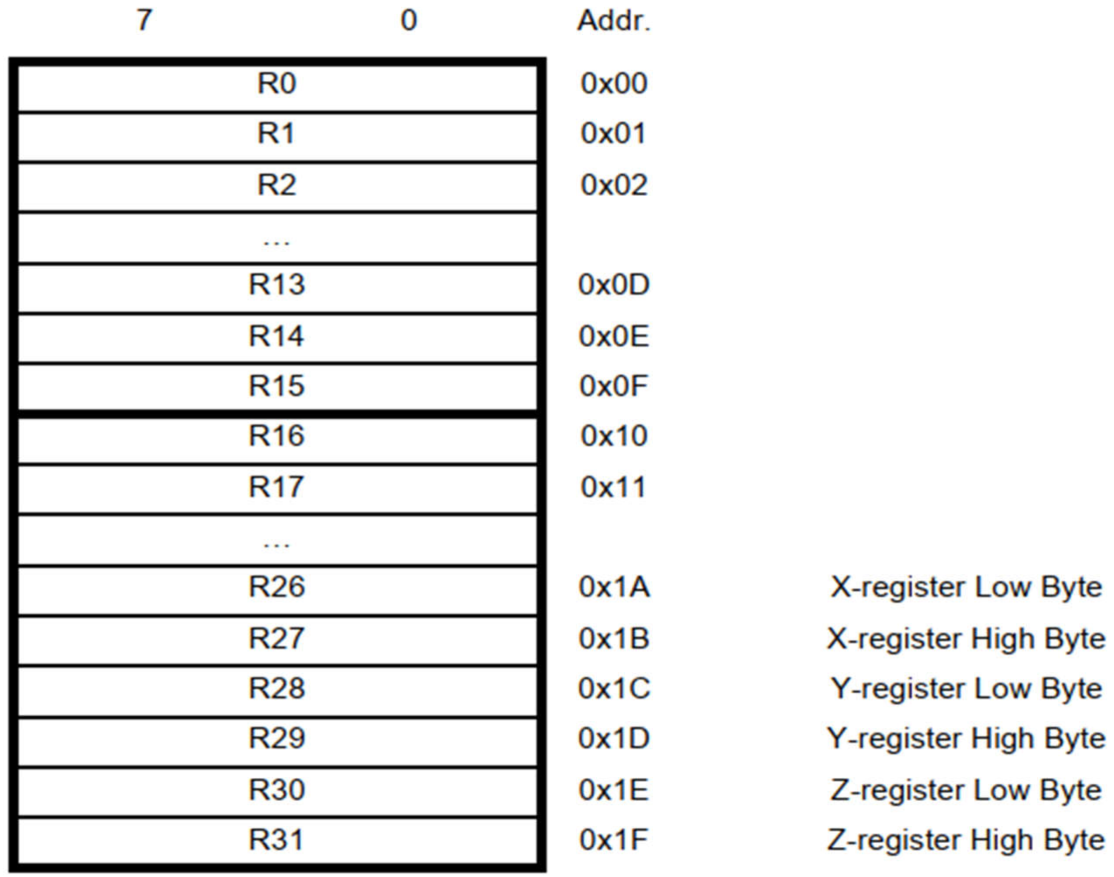
- **Bit 0 – C: Carry Flag**
  The Carry Flag C indicates a Carry in an arithmetic or logic operation.
  E.g: ADD, ADC (addition of two registers with carry), SUB (subtract two registers, SUBI(subtract constant from register) etc.

General Purpose Register File

In order to achieve the required performance and flexibility, the following input/output schemes are supported by the Register File:

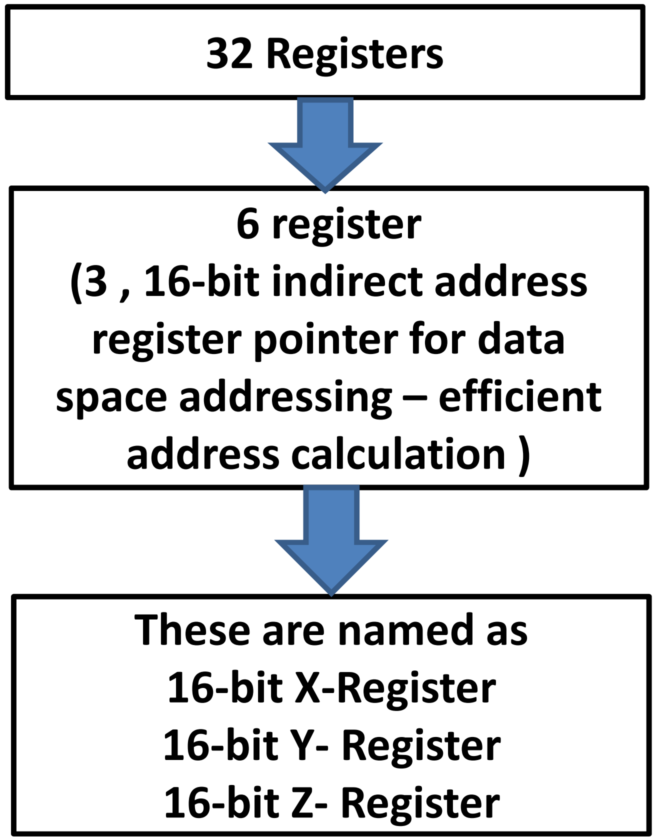
- One 8-bit output operand and one 8-bit result input
- Two 8-bit output operands and one 8-bit result input
- Two 8-bit output operands and one 16-bit result input
- One 16-bit output operand and one 16-bit result input

Structure of 32 General purpose working registers



7 0	Addr.	
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

- **General Purpose Registers :**



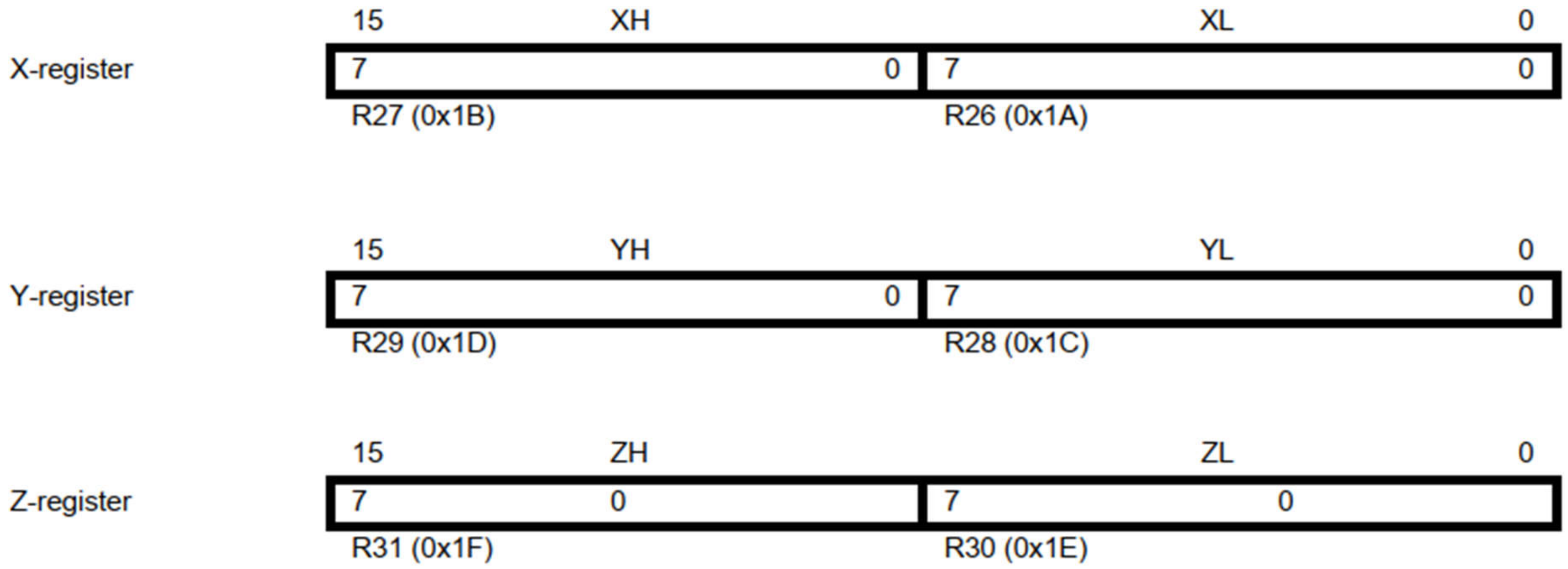
32 Registers

6 register
(3 , 16-bit indirect address register pointer for data space addressing – efficient address calculation)

These are named as
16-bit X-Register
16-bit Y- Register
16-bit Z- Register

X-Register, Y-Register, Z-Register (16-bit)

- The registers R26..R31 have some added functions (fixed displacement, automatic increment, and automatic decrement) to their general purpose usage. These registers are 16-bit address pointers for indirect addressing of the Data Space.


15
XH
XL
0
X-register
7
0
7
0
R27 (0x1B)
R26 (0x1A)
15
YH
YL
0
Y-register
7
0
7
0
R29 (0x1D)
R28 (0x1C)
15
ZH
ZL
0
Z-register
7
0
7
0
R31 (0x1F)
R30 (0x1E)